



**LEONTIEF METAMODELING FOR
MILITARY STRATEGIC EFFECTS**

THESIS

Anthony W. Snodgrass, Captain, USAF

AFIT/GOR/ENS/00M-20

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DTIC QUALITY INSPECTED 4

20000613 084

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE April 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE LEONTIEF METAMODELING FOR MILITARY STRATEGIC EFFECTS			5. FUNDING NUMBERS	
6. AUTHOR(S) Anthony W. Snodgrass, Captain, USAF				
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB OH 45433-7765			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GOR/ENS/00M-20	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) USAF/XOCA Attn: Maj Langbehn 1480 Air Force Pentagon Washington DC 20330-1480 DSN: 425-5061			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES Lt Col Mark A. Gallagher, USAF, ENS, DSN 785-6565 x 4335, Mark.Gallagher@afit.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
ABSTRACT (Maximum 200 Words) In this research, we propose and demonstrate an approach to incorporate currently un-modeled interactions into an existing or a new combat model or simulation. We apply the Leontief Input-Output model, which is a macro-economic model that measures the interactions between major industrial sectors of an economy, as a metamodel that periodically assesses the status of various combat units and support functions. If needed resources are insufficient, appropriate degrades are input into the combat model. As a result, dependencies that are not in the combat model are incorporated. Hence, we can assess the strategic or cascading effects of destroying a set of targets. We demonstrate our approach with THUNDER, the Air Force's campaign model. We estimated the Leontief model parameters with preliminary THUNDER runs. Some of the strategic targets—which formerly had no impact on the scenario—were connected in the Leontief metamodel. We ran the scenario again with degrades from the Leontief model being passed to THUNDER at regular intervals. This work provides a deeper understanding of the military structure as a system. Using the Leontief model as a metamodel is a promising approach for measuring the cascading impact of strategic targets on the entire military system.				
14. SUBJECT TERMS			15. NUMBER OF PAGES 109	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the U. S. Government.

AFIT/GOR/ENS/00M-20

LEONTIEF METAMODELING FOR MILITARY STRATEGIC EFFECTS

THESIS

Presented to the Faculty

Department of Operations Research

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Anthony W. Snodgrass, B.S., M.S.

Captain, USAF

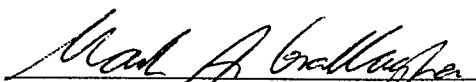
March 2000

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

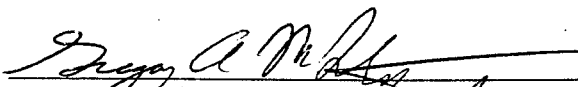
LEONTIEF METAMODELING FOR MILITARY STRATEGIC EFFECTS

Anthony W. Snodgrass, B.S., M.S.
Captain, USAF


Approved:


Lt Col Mark A. Gallagher (Advisor)

14 Mar 00
Date


Lt Col Gregory A. McIntyre (Reader)

14 MARCH 2000
Date


LTC Jack M. Kloeber Jr. (Reader)

14 March 2000
Date

Acknowledgements

I would like to express my appreciation to my advisor, Lt Col Mark Gallagher who has been both an intellectual and spiritual inspiration during the thesis process. I would also like to express my great thanks to Lt Col McIntyre whose expertise with THUNDER made the collection and reporting of results as painless as possible.

Steve Hadley and the S3I group have also been very helpful with preparing and debugging a one-off version of THUNDER 6.6 for this thesis. Without their patience and quick turn around times with replacement software as I “pushed the envelope again,” the project would have never gotten off the ground. I would also like to thank the group of instructors and students who met on Friday mornings for the free-for-all we called basketball.

Most importantly I want to recognize my wife Beth, who has kept the household intact and encouraged me at every turn. I would also like to thank Tyler and Allison. Their young smiles helped to break the tension and keep life in perspective.

Anthony W. Snodgrass

Table of Contents

Acknowledgements.....	v
List of Figures.....	vii
List of Tables.....	vii
1. INTRODUCTION.....	1
2. LEONTIEF INPUT-OUTPUT MODEL	7
2.1..... BASIC MODEL	7
2.2..... MODEL ASSUMPTIONS	12
2.3..... THE CONSUMPTION POSSIBILITY SCHEDULE	15
2.4..... THE DYNAMIC MODEL	20
3. ESTIMATING THE LEONTIEF MODEL FROM COMBAT DATA.....	29
3.1..... DEVELOPING THE INPUT-OUTPUT MODEL	29
3.2..... ESTIMATING THE LEONTIEF MODEL	32
3.3..... APPLYING THE LEONTIEF METAMODEL	36
4. THE METAMODEL OF THUNDER	40
4.1..... INTRODUCTION TO THUNDER	40
4.2..... THE SCENARIO	42
4.3..... ESTIMATING THE LEONTIEF MODEL	43
4.4..... THE LEONTIEF METAMODEL	46
4.5..... RESULTS OF STRATEGIC EFFECTS THROUGH LEONTIEF METAMODEL	50
4.6..... AN EXPERIMENT TO AID POLICY DECISIONS	54
5. CONCLUSIONS AND FOLLOW-ON RESEARCH.....	58

Bibliography.....	61
Appendix A: Leontief Input-Output Data Generation Program.....	62
Appendix B. Lingo Formats for Estimating Input Coefficients.....	65
Appendix C. Changes Made in THUNDER Databases.....	67
Appendix D. Interface and Solver Files.....	70

List of Figures

Figure 1. Clausewitz Trinity	3
Figure 2. Implication of Constant Returns to Scale.....	14
Figure 3. The Consumption Possibility Curve (Fuel=300).....	19
Figure 4. ME Scenario Situation Map	47
Figure 5. Effects from Destruction of Communications Center.....	52
Figure 6. Degrades due to Command Bunker Destruction.....	54
Figure 7. Use of a Control Variate.....	56
Figure 8. Comparing Initial Degrades Against All Target Sets.....	57

List of Tables

Table 1. Notional Leontief Input-Output Consumptions.....	8
Table 2. Variable Definitions.....	9
Table 3. Technical or Input Coefficients	9
Table 4. Notional Dynamic Model	23
Table 5. Solution for Dynamic Input-Output System.....	26
Table 6. Basic Leontief Model for Estimation.....	30
Table 7. Input Coefficient Matrix for Table 6	30
Table 8. The Leontief Inverse of Table 7	31
Table 9. Primary Factors Matrix pA	31
Table 10. Estimation of Input Coefficients.....	35
Table 11. The Estimated Leontief Table.....	36
Table 12. Estimated Leontief Model With Added Sector.....	37
Table 13. Results of Attacking Intel	38
Table 14. Originally Estimated Ground Commands.....	47
Table 15. Input Coefficient Matrix Used.....	48
Table 16. Comparison of Confidence Interval (CI) vs Control Variate (CV)	57
Table 17. Blue Aircraft Squadrons Deleted.....	67
Table 18. Configuration of Red Commands.....	67

Abstract

We define military strategic effects (also called cascading effects) as the direct and indirect impact of attacking a set of targets on the enemy's ability to wage war. We contend that military strategic effects result from interdependencies of major commands and weapons systems. By attacking critical areas of the military structure, an enemy's synergism can be destroyed.

Many combat models and simulations fail to account for all the interactions necessary to assess strategic effects. As the fidelity of combat model entities increases, the detailed interactions between various military units and support functions becomes more difficult to include. Using models that fail to capture strategic effects, however, can lead to decisions that do not value the true impact of weapon systems that can produce strategic results.

In this research, we propose and demonstrate an approach to incorporate currently un-modeled interactions into an existing or a new combat model or simulation. We apply the Leontief Input-Output model, which is a macro-economic model that measures the interactions between major industrial sectors of an economy, as a metamodel that periodically assesses the status of various combat units and support functions. If needed resources are insufficient, appropriate degrades are input into the combat model. As a result, dependencies that are not in the combat model are incorporated. Hence, we can assess the strategic or cascading effects of destroying a set of targets.

We demonstrate our approach with THUNDER, the Air Force's campaign model. We estimated the Leontief model parameters with preliminary THUNDER runs. Some

of the strategic targets—which formerly had no impact on the scenario—were connected in the Leontief metamodel. We ran the scenario again with degrades from the Leontief model being passed to THUNDER at regular intervals. This work provides a deeper understanding of the military structure as a system. Using the Leontief model as a metamodel is a promising approach for measuring the cascading impact of strategic targets on the entire military system.

**Leontief Metamodeling
for
Military Strategic Effects**

1. Introduction

With the increased cost of weapons systems, modeling and simulation has an expanding role in supporting defense decisions. For example, Congress legislated that each Presidential Administration report their assessment of the Department of Defense (DoD) at the beginning of each four year term. As a result, the Secretary of Defense directs the Quadrennial Defense Review (QDR) to analyze alternative force structures and provide a budgetary framework for the Administration's defense program. An urgent challenge faces the modeling and simulation community to support defense leaders in major decisions like the QDR. The DoD's current suite of military combat models fail to account for strategic effects, a significant aspect of military power. Military strategic effects may be defined as the synergistic, direct and indirect, impact on an enemy from attacking a set of targets. Weapon systems capable of delivering strategic effects are undervalued in current models and may go under funded as a result.

The Air Force is particularly interested in improving DoD's models to capture strategic effects because the value of airpower missions is generally measured indirectly. Air Force doctrine and air power advocates are not shy about this shortfall. A draft of the Air Force Doctrine Document 2-1 states, "Failure to properly analyze the mechanism that ties tactical results to strategic effects has historically been a failing of both airpower theorists and strategists"(AFDD 2-1, pg 3). Phillip Meilinger adds, "The challenge for airmen is to devise methods of analyzing the relationships between complex systems

within a country, determine how best to disrupt them, and then measuring the cascading effects of a system's failure throughout an economy"(Meilinger, pg 26).

As computing power has become relatively inexpensive, most of the modeling and simulation community has moved to highly detailed models in attempts to better model combat. The new high fidelity models often integrate components of lower level models from several different disciplines to build impressively complex systems driven by large quantities of data. Extensive linking and interaction of high fidelity components are difficult to develop and expensive to maintain. Often the result in many combat models is a stovepipe effect with limited interactions between different units, commands and various support functions. For example, ground and air commands maneuver independently and efficiently in simulations without the need for centralize command and control. These interactions must be accounted for to assess military strategic effects.

This research is based on the premise that military strategic effects are possible because of the interdependence of major commands and weapons systems. We contend that by attacking critical areas of the military structure, the synergism that existed in the original system can be destroyed and the total effect of the attacks are greater than the value of the individual targets. We propose and demonstrate a metamodel to incorporate strategic effects into existing or new combat models and simulations.

Before going any further, a detailed explanation of how we intend to measure military strategic effects is in order. The draft of AFDD 2-1 gives the definition of strategic effects as, "...the disruption of the enemy's strategy, ability, or will to wage war or carry out aggressive activity through destruction or disruption of their COGs (Centers

of Gravity) or other vital target sets, including command elements, war production assets, fielded forces, and key supporting infrastructure" (AFDD 2-1, pg 7).

Carl Von Clausewitz developed the concept of a COG between 1816 and 1830.

Clausewitz states:

What the theorist has to say here is this: one must keep the dominant characteristics of both belligerents in mind. Out of these characteristics a certain center of gravity develops, the hub of all power and movement, on which everything depends. That is the point against which all our energies should be directed. (Clausewitz, pg 595-596)

Clausewitz describes intrinsic principles and the systems they govern in war:

War is more than a true chameleon that slightly adapts its characteristics to the given case. As a total phenomenon its dominant tendencies always make war a paradoxical trinity—composed of primordial violence, hatred, and enmity, which are to be regarded as a blind natural force; of the play of chance and probability within which the creative spirit is free to roam; and of its element of subordination, as an instrument of policy, which makes it subject to reason alone.

The first of these three aspects mainly concerns the people; the second the commander and his army; the third the government. (Clausewitz, pg 89)

Military theorists have combined these passages to develop a model the Air Force

Doctrine Center questions as the center of gravity, shown in Figure 1 (AFDC, Slide 6).

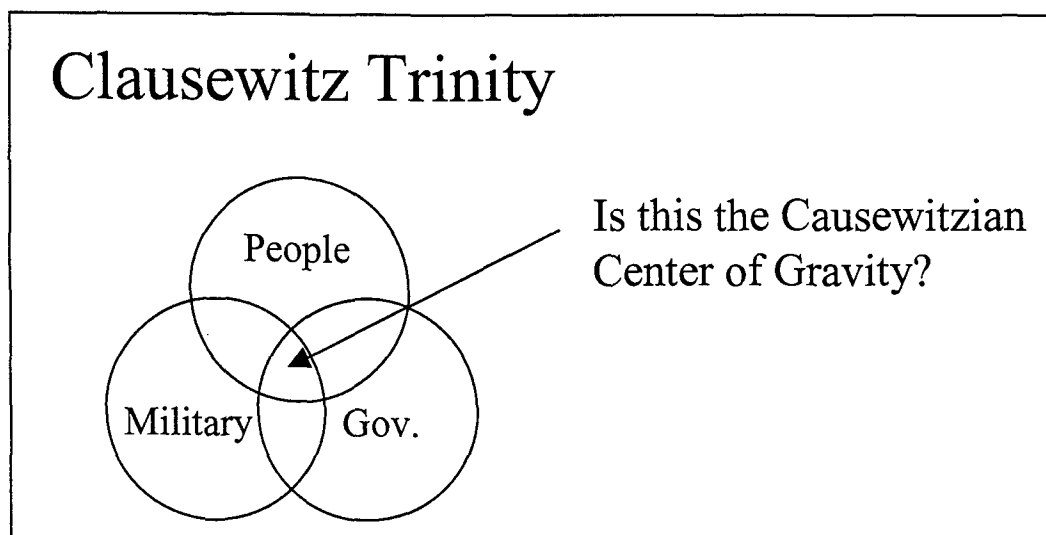


Figure 1. Clausewitz Trinity

A more current theorist by the name of John Warden argues that, “The term ‘center of gravity’ is quite useful in planning war operations, for it describes that point where the enemy is most vulnerable and the point where an attack will have the best chance of being decisive” (Warden, 1989: 7). Later he developed his strategy more fully by introducing the equation

$$(\text{Physical}) \times (\text{Morale}) = \text{Outcome}.$$

He contends the physical system is knowable and, by attacking it systematically, can be driven to near zero thus neutralizing the effect of morale on the outcome of war (Warden, 1995: 43). His system of analyzing the enemy for centers of gravity consists of a series of concentric rings with Leadership at the center followed by Organic Essentials, Infrastructure, Population, and Fielded Military in the outside ring. He argues that centers of gravity exist at each level and must be struck in parallel to generate strategic paralysis and force the enemy to capitulate (Warden, 1995:54). The importance of Warden’s work is still hotly debated. David Mets argues that there is little new in Warden’s theory (this is not meant derogatorily—Mets also states there was little new theory in Mahan’s *The Influence of Seapower* which does not diminish its importance). Mets further states Warden’s true contribution lies in the change of doctrine suggesting that the air forces can sometimes be the supported element while ground and naval forces are the supporting elements (Mets, 1998: 77). All of these theories describe the interaction of systems that should be incorporated in models of warfare.

We intend to measure military strategic effects by first focusing only on the military structure. We apply a model that captures the **dominant characteristics** of a belligerent’s system and their interactions. We assess system dependencies with the

Leontief Input-Output model, a macro-economic model that measures the interaction between major industrial sectors. By estimating a system's characteristics with an existing combat model, we focus on the **knowable physical** effects to determine a target set that **disrupts the enemy's ability to carry out aggressive activity**.

The Leontief model, as used in this research, is a metamodel that modifies the combat model. Periodically, the combat model is stopped, and the system's characteristics are passed to the Leontief model. Leontief degrades, resulting from dependencies previously not in the combat model are passed back to revise the systems' status. Hence, the additional strategic effects from direct and indirect interactions are incorporated into the combat model. We quantify the additional strategic effects as the difference between the results of the original model and the results with the Leontief metamodel.

A significant side effect of modeling strategic effects in this manner is the black box of the combat model becomes more transparent. This process begins by estimating a Leontief model that represents the sector interactions in the combat model. Because the Leontief model estimates the structure of the underlying system, the relationship of inputs, assumptions, and outputs become clear. Furthermore, aggregated analysis, such as sensitivity analysis, may be conducted using the Leontief model, rather than the combat model.

The following chapter describes the basic structure of the static and dynamic Leontief input-output model and discusses its applicability to combat modeling. Chapter Three details our methodology to estimate the Leontief model with data from a combat model that lacks strategic effects. Chapter Four presents a demonstration of the approach

to the Air Force's premier campaign model, THUNDER. Chapter Five concludes this research and suggests other potential applications.

2. Leontief Input-Output Model

Wassily Leontief earned the 1973 Nobel Prize in Economics for development and application of the input-output model. The value of the input-output model is that it captures the interactions between major sectors of an economy and is tractable. The model may be highly aggregated, such as those used for system of national accounts, or de-aggregated down to the level of a single manufacturer. As such, many analysts have applied the Leontief input-output model to study a variety of systems such as energy usage, environmental pollution, and employment in industry (Miller, 1985).

In this chapter, we discuss the basic Leontief Input-Output model, model assumptions, consumption possibility curves, and the dynamics model, respectively, in the following sections.

2.1. Basic Model

The static input-output model starts with a table of interactions between the sectors being modeled. Table 1 provides an aggregated notional example of military sectors and their consumption. It shows the relationship between an Air Force sector that generates sorties and an Army sector defending area with the quantities stated in daily requirements. In this table, reading down the first column reveals that the Air Force requires 25 sorties to defend itself from air attack, which fit into the category of Combat Air Patrol (CAP), 15 square miles of Army defense from missile attacks and 200,000 gallons of fuel. The outputs are read across the row and reveal that the Air Force generates 25 sorties to defend itself, 65 sorties to support the Army that may be considered Close Air Support (CAS), and 85 attack sorties for a total of 175. The inputs

and the outputs for the Army can be determined in a similar manner reading down the second column and across the second row.

Table 1. Notional Leontief Input-Output Consumptions

	Air Force	Army	Attacks	Total
Air Force (Sorties)	25	65	85	175
Army (Sq. Miles)	15	25	125	165
Fuel (Gallons x1000)	200	100		300

We relate the sector consumptions to the model parameters in the following way.

The total output of sector i is represented by X_i , and x_{ij} represents the amount of sector i outputs consumed by sector j . The amount of input from sector i delivered to a final demand sector is denoted by y_i . The total amount of primary resources available is represented by P_i and p_{ij} represents the amount of resource i consumed by sector j .

The output of sector i consumed by sector j **per unit of sector j 's total output** is called the input or technical coefficient, a_{ij} . A similar resource coefficient, r_{ij} , specifies the amount of resource i consumed per unit of sector j 's output.¹ The formulas for a_{ij} and r_{ij} are

$$a_{ij} = \frac{x_{ij}}{X_j} \quad r_{ij} = \frac{p_{ij}}{X_j} \quad (2-1)$$

For the notional example, shown in Table 1, define X_1 as Air Force Sorties, X_2 as Army Square Miles, and P_1 as fuel. Table 2 shows the variable definitions discussed above.

¹ The notation for the input coefficient of primary resources, r_{ij} , and total primary resources, P_i , is different from that of economic journals. In most economic journals there is only one primary resource and it is labeled X_0 . The notation is changed here since we will be using multiple primary resources throughout the rest of the paper.

Table 2. Variable Definitions

$x_{11} = 25$	$x_{12} = 65$	$y_1 = 85$	$X_1 = 175$
$x_{21} = 15$	$x_{22} = 25$	$y_2 = 125$	$X_2 = 165$
$p_{11} = 200$	$p_{12} = 100$		$P_1 = 300$

Next we convert these consumptions into rates of consumption per unit output as shown in Table 3.

Table 3. Technical or Input Coefficients

$a_{11} = \frac{25}{175} = 0.143$	$a_{12} = \frac{65}{165} = 0.394$
$a_{21} = \frac{15}{175} = 0.086$	$a_{22} = \frac{25}{165} = 0.152$
$r_{11} = \frac{200}{175} = 1.143$	$r_{12} = \frac{100}{165} = 0.606$

The column vector $[0.143, 0.086, 1.143]^T$ indicates for each additional output of X_1 , which is Air Force Sorties, consumes 0.143 of X_1 , 0.086 of X_2 , which is Army Square Miles, and 1.143 of P_1 , which is thousands of gallons of fuel. The 14.3 percent of sorties consumed to generate sorties represent the Counter Air Patrol (CAP). Similarly, the column vector $[0.394, 0.152, 0.606]^T$ associated with additional Army Square Miles Controlled indicates the consumption of X_1 , X_2 , and P_1 for each additional unit of X_2 produced.

Notice that a_{12} is the number of Close Air Support (CAS) sorties divided by the total number of square miles the Army defends, which gives the rate the Army requires

CAS to maintain ground control measured in square miles. In contrast, the a_{11} is simply a percentage of CAP sorties to the total USAF sorties.

The general form of a Leontief Input-Output model with m sectors and n primary inputs is the following set of linear equations

$$\begin{aligned}
 (1 - a_{11})X_1 - a_{12}X_2 - \cdots - a_{1m}X_m &= y_1 \\
 -a_{21}X_1 + (1 - a_{22})X_2 - \cdots - a_{2m}X_m &= y_2 \\
 &\vdots \\
 -a_{m1}X_1 - a_{m2}X_2 - \cdots + (1 - a_{mm})X_m &= y_m \\
 &\vdots \\
 r_{11}X_1 + r_{12}X_2 + \cdots + r_{1m}X_m &= P_1 \\
 &\vdots \\
 r_{n1}X_1 + r_{n2}X_2 + \cdots + r_{nm}X_m &= P_n
 \end{aligned}
 \tag{2-2}$$

where y_j is the direct consumption of sector j 's output. The set of $m+n$ equations are derived from the input-output table. Since a_{ji} represents the percentage of sector i output consumed within that sector, the diagonal terms generate the percent of gross output available for other sectors to consume. The terms a_{ij} represent the percentage of sector i output consumed in sector j .

For the notional example, the linear equations based on (2-2) and Table 3 are

$$\begin{aligned}
 (1 - 0.143)X_1 - 0.394X_2 &\geq 85 \\
 -0.86X_1 + (1 - 0.152)X_2 &\geq 125 \\
 1.143X_1 + 0.606X_2 &\leq 300
 \end{aligned}
 \tag{2-3}$$

Two important things to notice about (2-3) are that the equality signs have been replaced with inequality signs and that the third equation is different from the rest. The inequality signs make sense if you read the first equation in the following manner: the net output of

sector X_1 minus the input required for sector X_2 must be greater than or equal to the number of attack sorties that are directly consumed (offensive). The inequality in this case represents inefficiency in the system. This relaxation is necessary to provide a feasible region common in linear programming. The last equation is of special importance to the military because it deals with primary inputs such as logistics. It may be read: the fuel consumed by sector X_1 plus the fuel consumed by sector X_2 must be less than or equal to the total fuel available.

Any linear programming software package can determine the optimal allocation of resources for this problem. In addition, this formulation allows for a great deal of flexibility. For example, if only 200 thousand gallons of fuel are available in the notional problem, it is infeasible. Therefore, the limited fuel must be allocated to meet the commander's priorities. Multiple objective functions may be developed to reflect the mission objectives and generate the "best" solution given limited resources.

The use of fuel as a primary input separates this model from the typical economic model. Most economic models consider labor and land as primary inputs used in production—inputs that are rented but not diminished by production, as fuel would be. In economic models the amount of land or labor is considered to be constant from period to period and only the rents may change. Because the amount of fuel is diminished in each time period by production, a system of re-supply and accounting must be maintained outside of the model. This distinction will cause the military model to have a complex system to account for the amount of available resources.

The Leontief input-output model can be an open or a closed system. In the open system the column vector of final demands, y , is exogenous to the model and must be

provided. In economics, household consumption often serves as the vector of final demands. Then m sector equations and n equations of limited primary resources determine the value of the m sector outputs to satisfy those demands. In the closed model, one sector acts as the final consumer and provides the feedback of primary resources into the system. The vector of final consumption then is a zero vector and there are again m equations to solve for m unknowns with no primary resources in the system. In the closed system presented later, households account for all final consumption and supply labor as the feedback into the system. Our notional example and application uses the open model.

2.2. Model Assumptions

The model as described above incorporates three important assumptions: fixed coefficients of production, constant returns to scale, and homogeneity of input resources. The following paragraphs discuss each of these assumptions in some detail.

The concept of fixed coefficients of production is the strongest assumption of the model and based on the concept of efficiency. It is assumed that there exists some minimal input from each sector (for some sectors this is zero) in order to produce one unit of output. The model above assumes that 200,000 gallons of fuel is the minimum amount of fuel required to fly 175 sorties. Obviously, the same amount of sorties can be flown with 250,000 gallons of fuel but the extra fuel will not be wasted as long as fuel has value. In economic terms, an item that has no cost associated with it is considered a free good and in almost all analyses the free good will be over-consumed. In a combat scenario the input resources consist of logistics. Logistics is normally a limiting factor of

combat and will only move necessary resources to a combatant. The assumption that input resources have a cost associated with them and have value is valid using this logic. The absence of free goods leads us to our input coefficients previously denoted by a_{ij} in the following way. Assume that a_{ij} is the minimum input per unit from sector i to sector j . Then the following relationship exists:

$$X_1 \leq \min\left(\frac{x_{11}}{a_{11}}, \frac{x_{21}}{a_{21}}, \frac{p_{11}}{r_{11}}\right) \quad (2-4)$$

Notice that X_1 must be less than or equal to each of the ratios with the equality holding for only one limiting constraint. This implies that,

$$x_{11} \geq a_{11}X_1, \quad x_{21} \geq a_{21}X_1, \quad \text{and} \quad p_{11} \geq r_{11}X_1.$$

Dorfman states that in any system not involving free goods, the equality will hold for all ratios. (Dorfman, pg 210) The assumption of no free goods made earlier implies all of the ratios from (2-4) will hold as equalities and the system will be economically efficient. This relationship allowed us to divide every input in the first column by the total output for sector 1 to get the input coefficients of equation (2-1). For convenience, the values of the first column of the model from Table 1 are placed in the equation below. The ratio of each is equal to 175 (with rounding).

$$175 = \min\left(\frac{25}{0.143}, \frac{15}{0.086}, \frac{200}{1.143}\right)$$

The second assumption is a result from the concept that fixed coefficients implies constant returns to scale. The input-output model assumes that given twice the input resources, twice the output will result. Economists normally assume variable marginal

returns associated with increasing returns to scale from specialization. Near maximum production capacity, diminishing marginal returns result in less output for each additional unit of input and eventually results in diminishing returns. Figure 2 depicts variable marginal returns on the left. On the right is the same graph fitted with a constant returns to scale model including a maximum capacity associated with it. With this assumption, Leontief found good approximations with useful results.

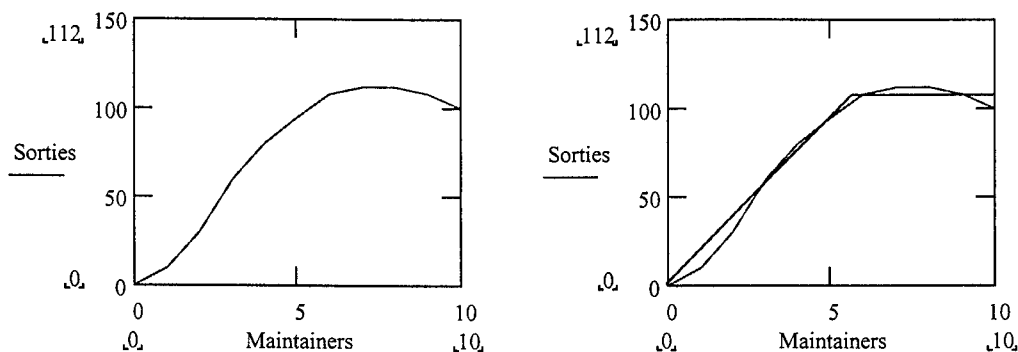


Figure 2. Implication of Constant Returns to Scale

The linear estimation often provides very useful results in the area of typical operating capacities (50 to 90%). The maximum operating capacity is normally associated with some type of physical constraint. For example, if an F-15 takes an average of four hours to fly a mission and two hours to reload and re-arm, it has a maximum capacity of four missions per day. The maximum constraint is simply another inequality added to the linear problem. The simplicity of the linear model along with its adequacy over the range of interest has resulted in the wide application of this approach. There are two major standards that maintain national input output accounts. The Bureau of Economic Analysis produces US statistics with input-output accounts as an integrating principle, and the System of National Accounts is an international standard used by the 29 countries

belonging to the Organization for Economic Co-Operation and Development. (Landefeld, p. 9) National Input-Output tables are available for over 60 countries.

Finally, the homogeneity of resources must be considered when building a model. In the notional example, the fuel was considered to be homogeneous since the same fuel was usable by both the Air Force and the Army. Aggregation of this type may be useful to the Joints Chiefs who are simply trying to account for the total quantity of fuel being consumed but it is of little use to logisticians trying to supply fuel to both sectors. The model also assumes that CAS, CAP and Attack sorties are all the same and the Army units protecting the Air Force base are the same as those being used for attack. In order to account for the difference between jet fuel and gasoline, air defense units and artillery, or F-15's and A-10's the table must be de-aggregated. We aggregate sectors to obtain macro-effects, but the approach may be applied at lower levels of aggregation where needed.

2.3. The Consumption Possibility Schedule

The consumption possibility schedule is a valuable tool often used in economics. The consumption possibility schedule illustrates the feasible final demands based on capacity constraints and limited primary factor resources. To derive the consumption possibility curves for the notional model discussed above, a restatement of the model in matrix notation is in order. The following notation contains equalities, but a generalization to inequalities as described above is not problematic. The model starts with m sectors and n primary inputs, which implies the following:

$$\mathbf{X} = \mathbf{aX} + \mathbf{y},$$

where \mathbf{X} is an m dimensional column vector of gross outputs for each sector, \mathbf{y} is an m -dimensional column vector that represents consumption, and \mathbf{a} is an $m \times m$ square matrix of input coefficients. The formula states that all inputs to each sector plus that used in final demand must equal total sector output and implies

$$\mathbf{X} - \mathbf{aX} = \mathbf{y} \text{ or } (\mathbf{I} - \mathbf{a})\mathbf{X} = \mathbf{y},$$

where \mathbf{I} is an identity matrix of size m . The equation most often seen in economic writing is

$$\mathbf{X} = \mathbf{Ay}, \tag{2-5}$$

where \mathbf{A} is equal to $(\mathbf{I} - \mathbf{a})^{-1}$. This formula allows the economist to determine the gross outputs required for a given vector of consumption \mathbf{y} .

Two conditions must exist for $(\mathbf{I} - \mathbf{a})$ to be invertible. First, the determinant of \mathbf{a} and all of its principle submatrices must be greater than zero. Second, $(1 - a_{ii})$ must be greater than zero for all i . These are known as the Hawkins-Simon condition. "The material interpretation of that condition is that if an economic system in which each sector functions by absorbing output of other sectors directly and indirectly is to be able not only to sustain itself but also to make positive delivery to final demand, each one of the smaller and smaller and smaller subsystems contained in it must necessarily be capable of doing so too." (Leontief, pg 26) In our notional example, this equates to the amount of sorties required to defend the base that generates an attack sortie, plus the number of sorties that go into army defense that goes into defending the base that generates attack sorties, cannot be greater than total attack sorties.

To complete the formulation, we must now consider our n primary factors. The formula for distributing the primary factors is

$$\mathbf{P} = \mathbf{r}\mathbf{X}, \quad (2-6)$$

where \mathbf{P} is the n dimension column vector of primary resources available to the economy and \mathbf{r} is an $n \times m$ matrix of input coefficients for the primary factors. You may verify that this is the last equation of (2-3) in our notional example. Substituting (2-5) for \mathbf{X} in the above equation produces

$$\mathbf{P} = \mathbf{r}\mathbf{A}\mathbf{y}. \quad (2-7)$$

We now have the information needed to develop a consumption possibility schedule for our notional example. The problem may be set in matrix notation as

$$\begin{bmatrix} 0.143 & 0.394 \\ 0.086 & 0.152 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix},$$

$$\begin{bmatrix} 1.143 & 0.606 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = P_1$$

where our number of sectors is equal to two and the number of primary inputs is one. In the form of equation (2-5), the problem becomes

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1.224 & 0.569 \\ 0.124 & 1.237 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$P_1 = 1.143X_1 + 0.606X_2 \quad (2-8)$$

The reader can confirm that by inserting the values of 85 and 125 for y_1 and y_2 the original total output values of 175, 165, and 300 (with minor rounding) are obtained.

Now performing the substitution from (2-5) we obtain

$$P_1 = 1.143(1.224y_1 + .569y_2) + .606(.124y_1 + 1.237y_2)$$

(2- 9)

In most economic journals, the above equation is simplified and presented as

$$P_1 = A_{31}y_1 + A_{32}y_2,$$

where the **A** matrix for the primary factors is simply **rA** from equation (2- 5). In our example, A_{31} equals 1.474 and A_{32} is 1.400 using the economic notation. Hence, the final possible output, y_1 and y_2 , are limited by the available primary input.

An important point from equation (2- 7) (which is easily recognized in (2- 9)) is the quantity of primary input for each unit of sector X_1 is accounted for in direct consumption from that sector, y_1 , and the indirect amount of X_1 used by other sectors to produce their final consumption, in this case y_2 . This accounting for both the direct and indirect consumption between sectors along with the implications of the Hawkins-Simon conditions is what makes the model so valuable. We contend this direct and indirect accounting is missing between some sectors in our combat models and leads us to believe it will also account for strategic effects.

It is now simple to produce a consumption possibility curve. From the last equation, when y_1 is zero, y_2 is equal to P_1/A_{32} and, similarly, y_1 is equal to P_1/A_{31} when y_2 is allowed to be zero. The graph illustrated in Figure 3 is generated using data from our example. The shaded area below the line is within our consumption possibilities given 300 thousand gallons of fuel. This consumption possibility schedule represents the range of strategies available to a commander in the notional model.

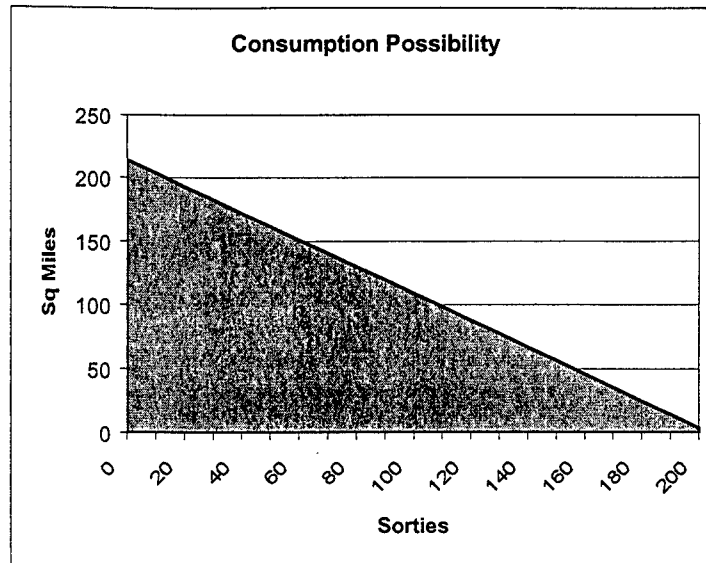


Figure 3. The Consumption Possibility Curve (Fuel=300)

If the strategy is to generate an Army offensive occupying 200 Square Miles ($y_2=200$), this requires 114 Air Force sorties and a total output of 250 Square Miles of Army occupation. Spare fuel would be available to fly another 13 attack sorties or may be saved for future use. The reader can confirm these numbers using the equations in (2- 8) above where the area under the curve is the result of inequality signs introduced earlier.

With only one primary factor, the consumption possibility curve is a straight line. In the more general case, there is a straight line associated with each primary factor and the consumption possibility appears as a convex polygon. This suggests that although we have used the assumption of constant returns to scale on our sectors, the consumption possibility will be non-linear as long as more than one primary factor is binding. Robert Dorfman also offers this explanation in the defense of constant returns to scale.

It is only when we insist on infinitesimal substitution, on a continuously varying marginal rates of transformation, on sensitivity of factor proportions to all price variations no matter how slight, that we have to give up the polygonal frontier for the neoclassical smooth curve. (Dorfman, pg 349)

The discussion so far has dealt only with the static model and has illustrated the value Dr. Leontief's model provides in our endeavor to capture strategic effects. In the next section the temporal aspect of the Leontief Input-Output model will be introduced, and the concept of the closed model will be demonstrated.

2.4. The Dynamic Model

In the previous section, time played no part in the model except in establishing a timeframe for building the original table. The static example showed that some sectors used its own inputs in order to produce output and the technical coefficient was labeled a_{ii} . In many cases, such as agriculture, these inputs are required from a previous time period's output (the seeds required for this year's crop must be from last year's harvest). In this section we will consider the dynamic implications of the Leontief system as it applies to the military. The section begins by looking at the Ramsey model and extending the lessons from this simple model to the dynamic Leontief model. Finally, the dynamic Leontief model will be considered in military terms.

Robert Dorfman introduces a simple linear dynamic model, called the Ramsey model, in which a single commodity uses only itself as an input at time t to reproduce a larger output at time $t+1$ (Dorfman, pg 267-269). The model considers the output for any time period, t , to be used as input for the next time period, $t + 1$. This input can be divided into consumption, $C(t+1)$, or stock, $x(t+1)$. Again, we will use a linear constant of a . The production function for any time period will be

$$C(t + 1) + x(t + 1) \leq ax(t) .$$

(2- 10)

For the above equation to be economically viable all of the variables must be greater than or equal to zero and a must be greater than one for a system that grows over time. The model is initiated with K units of our commodity so that when $t=0$;

$$C(0) + x(0) \leq K . \quad (2- 11)$$

At the next time period, $t=1$, the choice of consumption verses stock is

$$C(1) + x(1) \leq ax(0) \leq a[K - C(0)] . \quad (2- 12)$$

The extreme right side of this equation is simply (2- 11) solved for $x(0)$ and substituted.

The next time period produces

$$C(2) + x(2) \leq ax(1) \leq a[a(K - C(0)) - C(1)] . \quad (2- 13)$$

And in general the formula for any given time period is

$$C(n) + x(n) \leq a^n [K - C(0)] - a^{n-1}C(1) - \dots - aC(n-1) . \quad (2- 14)$$

If we choose a level of consumption in a given time period such as $C(3)$, we gain a needed insight on a dynamic system. First solve (2- 14) for the value of $C(3)$ to get

$$C(3) \leq a^3 [K - C(0)] - a^2C(1) - aC(2) - x(3) . \quad (2- 15)$$

By solving equation (2- 11) for $C(0)$, (2- 12) for $C(1)$ and (2- 13) for $C(2)$; then substituting into the above equation gives the following results:

$$\begin{aligned} C(0) &\leq K - x(0) \\ C(1) &\leq ax(0) - x(1) \\ C(2) &\leq ax(1) - x(2) \\ C(3) &\leq ax(2) - x(3) \end{aligned}$$

At every time period the required level of consumption is dependent on the stock available from the previous time. The stock, $x(2)$, must be large enough to produce $C(3)$ of consumption with any excess to be left for stock in the next time period, $x(3)$, but the amount of $x(2)$ was dependent on $C(2)$ and the stock $x(1)$ available and so on. **This suggests a dynamic system may be considered a series of static models related through the stock available from the prior period.**

With this concept in mind we now turn our attention to the dynamic input-output model capable of handling multiple interdependent commodities. As may be expected from the single commodity discussion, a method of handling stock from one time period to the next must be incorporated into the system. Leontief introduces the **B** matrix into a set of linear difference equations described by

$$\mathbf{X}(t) - \mathbf{aX}(t) - \mathbf{B}[\mathbf{X}(t+1) - \mathbf{X}(t)] = \mathbf{y}(t), \quad (2-16)$$

where **B** is an $m \times m$ matrix of capital coefficients, b_{ij} , that represent the amount of primary or intermediate capital goods produced by industry i that industry j must use per unit of its own output. Capital goods may be considered the stock on hand that a sector maintains as a part of normal operations. The rest of the variables are the same as those introduced in the static model with time coefficients added in the brackets².

The following example is taken from Dr. Leontief's second edition and demonstrates the closed and open dynamic model. (Leontief, pp 31-34) The closed system models an economy that has no imports or exports. In this example, households

² In Dr. Leontief's second edition the above equation is written with a capital A which he calls the matrix of input coefficients which is the **a** matrix introduced in the static model. The small **a** matrix notation will be maintained in this paper to avoid confusion.

represent the final consumption and, in return, provide labor as the primary resource.

The right hand side of equation (2- 16) now becomes column vector of zeros. In the open model, final consumption is considered exogenous to the system and must be provided or estimated through another system. An input-output table consisting of three sectors: Agriculture, Manufacturing and Households contain the relationship and units of measure listed in Table 2.

Table 4. Notional Dynamic Model (Leontief, pg 20)

	Agriculture	Manufacture	Households	Total Ouput
Agriculture	25	20	55	100 bushels
Manufacture	14	6	30	50 yards of cloth
Households	80	180	40	300 man years labor

We obtain the **a** matrix for the closed model as

$$a = \begin{bmatrix} 0.25 & 0.40 & 0.133 \\ 0.14 & 0.12 & 0.10 \\ 0.80 & 3.60 & 0.133 \end{bmatrix}.$$

The **B** matrix is provided as

$$B = \begin{bmatrix} 0.35 & 0.05 & 0.105 \\ 0.01 & 0.515 & 0.32 \\ 0 & 0 & 0 \end{bmatrix},$$

the columns of which represent the physical capital requirements per unit of output for each sector. In terms of our example, the third column represents the amount of agricultural products stored in household cupboards and textile products stored in closets.

The only restriction on the elements of this matrix is that they must be greater than or equal to zero.

The third row is of interest to the military considering combat applications. In this case the third row represents labor that cannot be stored from one period to the next generating zeros in this capital coefficient matrix. Unfortunately, this is also the case with most military sectors. Looking back at the static model used earlier, all of the military sectors except fuel have the same characteristic as they must be consumed in the same time period or their value is lost. While this may be a setback for military combat models, the rest of this example still holds promise for dynamic combat applications.

The closed model is able to answer the questions of how fast a system that reinvests based on consumption will grow and it's capability to maintain stability when faced with certain levels of consumption. The closed model is of little use in combat modeling where an opponent consumes our output, but it may be useful to those looking at the military industrial complex whose sole consumption and inputs come from the military sector. The questions of how fast can the military complex grow should we enter an extended war or what industries will fail based on budget reductions is of key strategic interest and insights may be obtained using the closed model. For the model in Table 4, Dr. Leontief reports that it may grow at the reciprocal of the largest characteristic root of the matrix $|\mathbf{I}-\mathbf{a}-\mathbf{B}|$ which he lists as $1/24.981$ or 4%, but we have not reproduced those results. While the closed model provides insight to certain characteristics of a system, Dr Leontief remarks, "Nevertheless, for the purposes of most practical applications, the closed version of the dynamic model has proved to be too deterministic and too rigid; the

input-output analysis is usually conducted in terms of the open version of the dynamic model...” (Leontief, pg 33). Following his wisdom, we will now turn our attention to the open model.

As mentioned earlier, the open model assumes that consumption (in this case Households) is exogenous to the model and; therefore, the inputs from Households back into the model are also considered unknowns. The **a** and **B** matrices for this model now become only the first two rows and columns of the matrices listed above representing a two equation system in terms of (2- 16). We are given the value of $X(0)$ and levels of consumption in each time period as

$$X(0) = \begin{bmatrix} 115 \\ 60 \end{bmatrix}, y(0) = \begin{bmatrix} 55 \\ 30 \end{bmatrix}, y(1) = \begin{bmatrix} 57.7 \\ 31.5 \end{bmatrix}, \text{ and } y(2) = \begin{bmatrix} 60.6 \\ 33.3 \end{bmatrix}$$

We may now generate the total output, investment, and level of employment necessary to achieve these goals.

The dynamic system must be solved using equation (2- 16) which in terms of $X(t+1)$ becomes

$$X(t+1) = B^{-1}[(I - a + B)X(0) - Y(0)].$$

(2- 17)

This may not be solvable if **B** is singular, since the only restriction on the contents of **B** is that they are greater than or equal to zero. Applying equation (2- 17), the rounded total output at $t+1$ becomes

$$\begin{bmatrix} 2.865 & -0.278 \\ -0.056 & 1.947 \end{bmatrix} * \left[\begin{pmatrix} 0.4 & -0.45 \\ -0.15 & 0.365 \end{pmatrix} * \begin{pmatrix} 115 \\ 60 \end{pmatrix} - \begin{pmatrix} 55 \\ 30 \end{pmatrix} \right] = \begin{bmatrix} 134 \\ 73 \end{bmatrix},$$

which is the total output at X(1). Inserting the total output at X(1) into the stock transformation formula, given by $B[X(t+1)-X(t)]$, results in the total capital stock investment for the period.

$$\begin{bmatrix} 0.35 & 0.05 \\ 0.01 & 0.515 \end{bmatrix} * \left[\begin{pmatrix} 133.9 \\ 72.6 \end{pmatrix} - \begin{pmatrix} 115 \\ 60 \end{pmatrix} \right] = \begin{pmatrix} 7.3 \\ 6.7 \end{pmatrix}.$$

In this case, the un-rounded values for total output were used to obtain the results given by Dr Leontief in the table below. Finally, total employment must be calculated in a manner similar to equation (2- 6) with each sector being calculated individually using the third row of the original **a** matrix. The results for the initial year, $t=0$, are: Agriculture employs $0.8*115=92$ man-years of labor, Manufacture $3.6*60=216$ man-years, and Households $0.133*(55+30)=11$ for a total employment of 319 man-years. Each year must be calculated in a similar manner to obtain the results in the following table.

Table 5. Solution for Dynamic Input-Output System (Leontief, pg 34)

Year	Final Demand		Total Output		Investment		Employment			
	Ag	Man	Ag	Man	Ag	Man	Ag	Man	Hshld	Total
0	55	30	115	60	7.3	6.7	92	216	11	319
1	57.7	31.5	134	73	13.6	13.7	107	261	12	380
2	60.6	33.3	169	99	26.8	29.9	135	355	12	502

The dynamic model does allow for over-consumption in a given time period. In the example above at time $t=1$ the total output was calculated to be 134 bushels of

agriculture and 73 yards of cloth. This total output generates a total consumable output of 71 bushels of agriculture and 45 yards of cloth. By using equation (2- 17) we can show that the economy could possibly consume 121 bushels of agriculture and 84 yards of cloth and still have a minimal positive output for time $t=2$. This is possible only because the entire stock of capital will also be consumed. The stock of capital available at time $t=1$ is given by $\mathbf{BX}(1)$ and is:

$$\begin{bmatrix} 0.35 & 0.05 \\ 0.01 & 0.515 \end{bmatrix} * \begin{pmatrix} 134 \\ 73 \end{pmatrix} = \begin{pmatrix} 50 \\ 39 \end{pmatrix}.$$

While this amount of consumption is possible, it nearly destroys the system as the total output possible in the next time period is 6 bushels of agriculture and 1 yard of cloth.

Hence, over consumption leads to later shortages.

Three items should be emphasized about the dynamic model. First is the accounting process the input-output models uses. In the last example, if we had calculated the total output using the static model (ignoring the change in stock levels), the result for $X(0)$ using equation (2- 5) would have been

$$\begin{bmatrix} 1.457 & 0.6623 \\ 0.2318 & 1.2417 \end{bmatrix} * \begin{pmatrix} 55 \\ 30 \end{pmatrix} = \begin{pmatrix} 100 \\ 50 \end{pmatrix}.$$

Having been told that the real total output of each sector was actually 115 and 60, one might casually (and erroneously) assume that 15 units of agriculture and 10 units of manufacturing had been invested in capital stock. Only by realizing that the additional capital stock is in the form of processed goods and not raw material do we find the true investment in stock by $\mathbf{X}-\mathbf{aX}=\mathbf{y}$ resulting in the correct answer as shown below.

$$\begin{pmatrix} 15 \\ 10 \end{pmatrix} - \begin{bmatrix} 0.25 & 0.4 \\ 0.14 & 0.12 \end{bmatrix} * \begin{pmatrix} 15 \\ 10 \end{pmatrix} = \begin{pmatrix} 7.3 \\ 6.7 \end{pmatrix}$$

The second point is the dynamic model may not be useful for combat models since the output of combat sectors (sorties, area defended) like labor above cannot be stocked. The **B** matrix of a combat model will contain all zeros, leaving the static model. This said, the dynamic model may be extremely useful for the logistician whose sectors would be fuel, ammunition, personnel, tanks, and equipment to name a few. The dynamic model appears ideally suited for prioritizing the shipment of logistics not only to satisfy the current input requirements of combat but also boosting the capital stocks necessary for an offensive or more intense operations at a future period of time. Finally, the dynamic input-output model is again solved using a series of static models tied together by the status of current inputs at each time period.

3. Estimating the Leontief Model from Combat Data

The discussion up to this point has assumed that all of the information to construct the Leontief model is available. Combat models differ from economic models in one aspect because all of the information is not available in combat models. In fact, the consumptions between sectors, x_{ij} 's, are the items of interest in combat models, but not available as output. This chapter examines the structure of the Leontief input-output system, establishes a step-by-step process to develop an input-output model with multiple primary resources, and introduces a method of estimating the system parameters using total sector output, which is the data readily available from most combat models.

3.1. Developing the Input-Output Combat Model

In order to demonstrate this methodology, we begin with a small model in which all of the parameters are known. The model generated here has two ground commands (GC1 and GC2) with outputs measured in square miles of area, a squadron of A-10 aircraft (Sq1), and a squadron of F-117 aircraft (Sq2) whose output is measured in sorties. From this known model, output data is generated which can be used to estimate the original parameters. The model shown in Table 6 is developed using an Excel spreadsheet. The four sectors of "production" are highlighted to show the core of the model. There are six primary resources that include tanks, artillery, A-10's, F-117's, fuel, and ammo. The resources show that the ground commands each have 150 tanks, Squadron 1 has 14 A-10's and Squadron 2 has 28 F-117's. Fuel and ammunition has been aggregated and are considered to be homogeneous resources.

Table 6. Basic Leontief Model for Estimation

	GC1	GC2	Sq1	Sq2	Attack	Total
GC1	5	2	10	0	60	77
GC2	2	5	0	10	75	92
Sq1	20	25	0	0	0	45
Sq2	0	0	0	0	5	5
Tanks	150	150	0	0		300
Artillery	75	75	0	0		150
A-10's	0	0	14	0		14
F-117's	0	0	0	28		28
Fuel	60	60	100	80		300
Ammo	75	75	35	15		200

The input coefficient as described by Equation (2- 1) can be generated from this table. Dividing each entry of each column by the total output from the associated row generates the coefficients. For example, each entry in column one is divided by the total output for the GC1 sector (shown in row one as 77). The matrix of input coefficients and the coefficients of the primary resources are shown in Table 7.

Table 7. Input Coefficient Matrix for Table 6

	GC1	GC2	Sq1	Sq2
GC1	0.06	0.02	0.22	0.00
GC2	0.03	0.05	0.00	2.00
Sq1	0.26	0.27	0.00	0.00
Sq2	0.00	0.00	0.00	0.00
Tanks	1.95	1.63	0.00	0.00
Artillery	0.97	0.82	0.00	0.00
A-10's	0.00	0.00	0.31	0.00
F-117's	0.00	0.00	0.00	5.60
Fuel	0.78	0.65	2.22	16.00
Ammo	0.97	0.82	0.78	3.00

Finally, the **A** matrix described in Equation (2- 5) was generated by taking the inverse of **(I-a)** and is provide in Table 8. This matrix includes only the four sectors of production in the model.

Table 8. The Leontief Inverse of Table 7

	GC1	GC2	Sq1	Sq2
GC1	1.14	0.10	0.25	0.20
GC2	0.03	1.06	0.01	2.12
Sq1	0.31	0.31	1.07	0.63
Sq2	0.00	0.00	0.00	1.00

From Table 8 and Equation (2- 5), total output data can be generated by randomly creating consumption values for sectors one, two, and four. Sector three's consumption will always be zero for the purposes of this experiment because we are assuming that it has only a support role. In contrast, sector four has only an offensive role and does not support any of the other sectors. Based on the Equation (2- 7), the amount of primary resources necessary for the production of total output can be created for each bill of goods to be consumed. A small program which performs the calculations discussed above and stores the data in an Excel worksheet was written in Visual Basic, and it can be found in Appendix A along with some of the sample data. For convenience, values of the matrix pA are presented in Table 9 below. We now have a set of data that is consistent with data we would receive from a deterministic combat model and may begin the business of estimating the original input coefficients.

Table 9. Primary Factors Matrix pA

	GC1	GC2	Sq1	Sq2
Tanks	2.28	1.92	0.51	3.84
Artillery	1.14	0.96	0.25	1.92
A-10's	0.09	0.10	0.33	0.20
F-117's	0.00	0.00	0.00	5.60
Fuel	1.59	1.47	2.58	18.93
Ammo	1.38	1.21	1.08	5.41

3.2. Estimating the Leontief Model Parameters

Assume for the moment the only information we get from our combat model is total output and the total amount of resources consumed for the given period of activity. For every set of consumption values we artificially generated we received a total output for each of our four sectors. These four values generate four linear equations of the form

$$\mathbf{X} - \mathbf{aX} = \mathbf{y}. \quad (3-1)$$

The following eight equations are generated from the first two lines of total output data from Appendix A.

$$\begin{aligned} 100 - a_{11}100 - a_{12}142 - a_{13}65 - a_{14}15 &= y_1 \\ 142 - a_{21}100 - a_{22}142 - a_{23}65 - a_{24}15 &= y_2 \\ 65 - a_{31}100 - a_{32}142 - a_{33}65 - a_{34}15 &= y_3 \\ 15 - a_{41}100 - a_{42}142 - a_{43}65 - a_{44}15 &= y_4 \\ 112 - a_{11}112 - a_{12}181 - a_{13}78 - a_{14}41 &= y_5 \\ 181 - a_{21}112 - a_{22}181 - a_{23}78 - a_{24}41 &= y_6 \\ 78 - a_{31}112 - a_{32}181 - a_{33}78 - a_{34}41 &= y_7 \\ 41 - a_{41}112 - a_{42}181 - a_{43}78 - a_{44}41 &= y_8 \end{aligned}$$

The bounds on the variables are not shown but each a_{ij} and y_i must be greater than or equal to zero. Notice there are sixteen input coefficients to be estimated plus a consumption coefficient for each equation. The fact that we get a new consumption coefficient with each new equation means there will always be more unknowns than there are equations. In order to estimate the input coefficients, an estimate for the consumption variables, y_i is required.³

The primary resources also may be estimated using the equation

³ A non-linear search of the solution space is another possible technique that was not selected due to the timeline of this project.

$$\mathbf{P} = \mathbf{X}\mathbf{r}.$$

Each repetition generates new information about the twenty-four primary input coefficients since both \mathbf{P} and \mathbf{X} are known.

There are two issues to be addressed at this point. The first is the use of inequalities to allow for inefficiencies in the system and second is selecting an objective function for parameter estimation. The idea of inefficiencies in the system was mentioned in Chapter Two. There, the logic was put forward that logistics is a limiting factor for the military and only valuable resources would be provided to the combatants resulting in no free goods and an efficient system. This would suggest the use of equalities as in (2- 2). In contrast, even if the resources are scarce and have value, the commanders may choose or be forced to operate at inefficient levels for other reasons. Surely combat models based on stochastic inputs will not always operate at peak efficiency. From this discussion we will adopt the criteria that while the combat system is not entirely efficient, it is nearly efficient. It is prudent at this point to examine the objective functions and consequences of using inequalities in estimating input coefficients.

Equation (3- 1) may be rewritten to take the form of

$$\mathbf{X}\mathbf{y} = \mathbf{a}\mathbf{X} + \mathbf{e}. \quad (3- 2)$$

where \mathbf{e} is a vector of error terms that must be greater than or equal to zero. When written in this form the model looks suspiciously like regression, and our method is a form of regression. Here, instead of having the objective of minimizing the sum of square errors, we simply want to minimize the sum of errors. This model fits the data as

well as possible. In economic terms, the system is as efficient as possible. Furthermore, by minimizing errors, we are requiring more interaction in the model. In other words, we assume the underlying combat model is integrated and force as much integration as possible.

This process is not without its flaws. Since the third sector has no direct consumption, each data point takes the form of

$$X_3 - a_{31}X_1 - a_{32}X_2 - a_{33}X_3 - a_{34}X_4 - e_{i3} = 0.$$

Hence, sectors without consumption form a homogenous linear system. All homogenous linear systems have the trivial solution of all zeros as a possible answer. With an objective of minimizing the excess, a_{33} is set equal to one and the resulting excess is equal to zero. Since the coefficient of X_3 is $1 - a_{33}$, this is the trivial solution. The sum of the excess values is then significantly understated and the values of other a_{3i} 's are zero. The problem is corrected by estimating the parameter with a_{33} set equal to zero to force a non-trivial solution. This is justified because sector three was assumed to be in a support role. Here we are simply saying that it does not support itself. The approach, while providing a larger objective value of excess, allows for a_{3i} 's to be estimated correctly.

The result of using this approach is described in Table 10. These estimates were obtained using Lingo with one hundred data points. The scripts used for the Lingo runs are contained in Appendix B. The table also contains information on the power of the model, which was derived by dividing the solution to the objective function (sum of errors) by the sum of total outputs minus total consumption and subtracting from one. Equation (3- 2) demonstrates that if the sum of the error terms is zero we have modeled the system with perfect accuracy. On the other hand, if the sum of error terms is equal to

X-y our model has no explanatory power. This power is similar to the R-square term in standard regression. Hence, these estimates explain 99.8 % of the variation in this simple data set.

Table 10. Estimation of Input Coefficients

	True Values from Model	Regression
a_{11}	0.06	0.12
a_{12}	0.02	0.08
a_{13}	0.22	0.0
a_{14}	0.0	0.0
a_{21}	0.03	0.02
a_{22}	0.05	0.05
a_{23}	0.0	0.0
a_{24}	2.0	2.00
a_{31}	0.26	0.26
a_{32}	0.27	0.26
a_{33}	0.0	0.0
a_{34}	0.0	0.0
a_{41}	0.0	0.0
a_{42}	0.0	0.0
a_{43}	0.0	0.0
a_{44}	0.0	0.0
Objective Value (sum of e 's)		150.16
Total Output-Consumption		62683
Power of Estimation		$1 - .0024 = .998$

The Leontief model, including the estimates of the primary resources, using 100 data points and the Lingo models in Appendix B is shown in Table 11. This table should be compared closely with Table 7 to note the effects of homogeneous resources on the system as a whole. It is interesting to note that the A-10's have actually been estimated as required items for GC1 and GC2; and Tanks and Artillery have been distributed to the two flying squadrons. The ground command's fuel requirements are also larger to account for the aircraft misallocation. This type of error could have been avoided using a

priori knowledge making $p_{13}, p_{14}, p_{23}, p_{24}, p_{31}, p_{32}, p_{41},$ and p_{42} equal to zero prior to estimation. To show that the estimations, even with the misallocated resources, are still very good approximations for the original model they will be left in the following example.

Table 11. The Estimated Leontief Table

	GC1	GC2	Sq1	Sq2
GC1	0.12	0.08	0	0
GC2	0.02	0.05	0	2
Sq1	0.26	0.26	0	0
Sq2	0	0	0	0
Tanks	1.747	1.42	0.751	0.006
Artillery	0.814	0.627	0.641	0.029
A-10's	0.072	0.079	0.015	0.005
F-117's	0	0	0	5.58
Fuel	1.066	0.978	1.034	16.006
Ammo	0.807	0.622	1.426	3.046

3.3. Applying the Leontief Metamodel

In an application, the test of how well the estimation fits the true model is to run the combat model using the Leontief estimates as a metamodel to find the number and magnitude of degrades passed back to the model. A degrade occurs when the metamodel reduces to the total output variables in the combat model and is indicated whenever the consumption values passed to the metamodel can no longer be supported. The metamodel may take the form of

$$\begin{aligned}
 &\text{Max } \sum y \\
 &\text{st: } \mathbf{X} - \mathbf{aX} \geq \mathbf{y} \\
 &\quad \mathbf{p}_{\text{data}} \geq \mathbf{rX} \\
 &\quad \mathbf{y} \leq \mathbf{y}_{\text{data}}
 \end{aligned}$$

(3- 3)

where y_{data} is the estimates of consumption from the combat model, P_{data} is the value of primary resources available from the combat model, \mathbf{a} and \mathbf{r} are the estimated parameters. Because the metamodel cannot upgrade sectors, there are also bounds that \mathbf{y} must be less than or equal to y_{data} . The solution for the maximum value of consumption, \mathbf{y} , now relies on the interactions of the total outputs, \mathbf{X} , which are limited by the available resources, P_{data} . A degrade is passed back to the combat model on any repetition where the objective function is less than the sum of y_{data} . These degrades need to be investigated to see if they significantly alter the outcome of the battle.

In the following example a new sector called Intel has been added to the input-output model. This is equivalent to adding links for un-modeled interactions in the combat model. A small amount of Intel is required as an input for GC1 and GC2, the A-10's for this example require no Intel, and the F-117's require a great deal of Intel. In return, the ground commands and F-117's provide input to the Intel sector. The initial condition for the Intel sector is at 100% or total output is given as 100 units. In this case the units are not important since the sector only serves to illustrate the effects of a system when Intel has been diminished. Table 12 shows the revised military system.

Table 12. Estimated Leontief Model With Added Sector

	GC1	GC2	Sq1	Sq2	Intel
GC1	0.12	0.08	0.0	0	0.01
GC2	0.02	0.05	0	2	0.01
Sq1	0.26	0.26	0	0	0
Sq2	0	0	0	0	0.025
Intel	0.03	0.03	0	0.25	0
Tanks	1.747	1.42	0.751	0.006	0
Artillery	0.814	0.627	0.641	0.029	0
A-10's	0.072	0.079	0.015	0.005	0
F-117's	0	0	0	5.58	0
Fuel	1.066	0.978	1.034	16.006	0
Ammo	0.807	0.622	1.426	3.046	0

Simulated attacks are made on Intel, and the sector is incrementally drawn down to 5% of its original total. The rest of the sectors total output remains constant for the experiment to isolate the effects of Intel on the rest of the system. The results of attacking the Intel sector alone using the model in Equation (3- 3) are shown in Table 13. The initial output vector for the original sectors is (115, 258, 100, 62) from our first data point in Appendix A. The Base column shows the attack capability by sector generated from the actual values of the original model. The attack capability for the Intel sector is actually the slack in the system since Intel would again be considered a support organization with no attack capability. The values for the primary resources are the amounts consumed, given the total output generated. The 100% column shows that just by adding the Intel sector we have used a small amount of attack capacity to support Intel.

Table 13. Sector Attack Capacities with Various Intel Output

Intel Output	Base	100%	75%	50%	30%	20%	10%	5%
GC1	80	80	80	80	80	80	80	26
GC2	117	117	117	117	117	117	117	117
Sq1	0	0	0	0	0	0	0	0
Sq2	62	61	61	61	61	41	10	0
Intel		68	43	18	0	0	0	0
Tanks	645.36	637.12	637.12	637.12	642.53	558.06	439.07	281.03
Artillery	322.68	317.68	317.68	317.68	320.85	278.19	218.74	139.17
A-10's	31.13	30.26	30.26	30.26	30.48	26.21	20.10	13.44
F-117's	347.20	345.96	345.96	345.96	344.38	230.13	55.35	0.70
Fuel	1472.53	1465.04	1465.04	1465.04	1465.41	1072.55	480.50	212.14
Ammo	586.52	578.37	578.37	578.37	582.60	466.70	298.37	172.44

The attacks on Intel only become binding when it is reduced to less than 30%. As expected, at this level the F-117's are the most affected reducing the number of effective sorties by 35% when Intel is diminished to 20% of its original capacity. As a result of the

original structure, which required two square miles of defense per F-117 sortie, output of GC2 may be maintained with far fewer resources when not defending the F-117's. In fact, the attack capability of the ground commands can be maintained until Intel is driven to 5% of its original capacity. At 5%, the remaining Intel is then directed into the most efficient user of resources (GC-2) while the other command begins to suffer. It is of interest to note that the amount of resources required to maintain the ground attacks continues to diminish due to the original structure of the problem.

This demonstration only illustrates degrades the metamodel would pass back to a combat model. It is assumed that these degrades would have further implications in the adjudication of the battle and the dynamics between the two models would explain the value of blue intelligence on the system as a whole. In the case described above, even if intelligence was left at the 20% level with the F-117's hamstrung, the red army could possibly remain stronger and the combat model would degrade the ground commands as a result.

In this chapter we have developed a methodology to estimate the Leontief model from a combat model and investigated some of the interesting wrinkles that result from the mathematics. In the small problem developed no *a priori* information was used and there were some errors in the estimation as a result. Even with those errors, the estimation was very accurate as shown in the first two columns of Table 13, when the added Intel sector is ignored. The next chapter will apply this methodology to the Air Force's premier theater combat model called THUNDER and results from that analysis will be presented.

4. The Metamodel of THUNDER

In this section we will briefly introduce THUNDER and discuss the basics of the scenario chosen for this proof of concept. In performing this experiment we will use the data from THUNDER to estimate the Leontief metamodel and then use the metamodel to drive degrades in THUNDER. Details of the transactions between THUNDER and the Leontief metamodel are presented. The interface developed by System Simulation Solutions, Inc. (S3I) will be discussed and an estimation of the metamodel will be demonstrated. Finally, the results of introducing two of the strategic targets from the scenario will be discussed and an experiment will be conducted to demonstrate how this methodology may be used to aid decision-making.

4.1. Introduction to THUNDER

The predecessor to THUNDER was a deterministic model called TAC Warrior, a theater level model used from the late 1970's through the middle 1980's. The Air Force Studies and Analysis Agency (AFSAA) undertook the development of THUNDER in 1983 to address the shortcomings of TAC Warrior. The first operational version of THUNDER was released with version 2.0 in 1986. New versions have been released every seven to fourteen months since then, and AFIT currently runs version 6.6. The contractor in charge of developing the model is S3I.

THUNDER is currently the Air Force's theater level analytical campaign simulation model. It is a two sided, stochastic simulation of interactions between the ground war, logistics, ISR, and the air war at a campaign level of detail. This data driven

model was designed to aid policymakers in evaluating military strategy and capabilities, force structure and operation effectiveness in a joint warfighting context. Like all analytical models, THUNDER provides more insights than answers. There are 99 data files that feed information to the simulation and, therefore; it is up to the analyst to develop credible data designed for the scenario in question.

One of the 99 files contains a prioritized list of blue and red strategic targets each air force is given to destroy. This list may include power plants, communications centers, command bunkers, etc. Most of these targets are not related to the rest of the combat scenario. As a result of these targets being isolated, they act as a sink for the air forces. Each air force is required to generate sorties to destroy the targets but no benefit is derived from its success and the opportunity to use that sortie in other areas with impact on the scenario outcome is lost. The example of this chapter will show how the Leontief metamodel can be used to tie two of the strategic targets into the outcome of the ground war.

The ground war in THUNDER is based upon the Army Concepts Analysis Agency (CAA) Concepts Evaluation Model (CEM). As units engage in combat along the forward line of troops (FLOT), combat is adjudicated by CAA's Attrition Calibration (ATCAL) model. In evaluating FLOT movement the strength of the unit is important in two ways. First, the strength of the unit is used to evaluate its posture. The posture of the unit varies from hasty defense to attack. Second, the strength of the unit is used to estimate that unit's attrition advantage over the unit on the opposite side of the FLOT. The posture, attrition advantage, terrain, and supplies available are then used to calculate

the movement of the FLOT. All FLOT movements are deterministically evaluated at the unit level.

THUNDER models air warfare with discrete-event, stochastic methods and addresses all air warfare elements including mission planning, base operations, base logistics, flight group assembly, flight group movement, etc. THUNDER drives the air war with air mission planning. The purpose of air mission planning is to create Air Tasking Orders (ATOs). Aircraft squadron sorties are allocated to certain missions using a linear program to maximize squadron effectiveness in terms of capability, lethality, and mission priority. Once the squadrons have been allocated to specific missions they are assigned to enemy targets based on target priorities. Each target is generated based upon the perceived state of enemy resources and then given a target priority.

4.2. The Scenario

For this proof of concept research, we have chosen the unclassified Middle East (ME) database supplied with the software. In this database, the red Iraqi army has nine commands with a total of fifty-one units assigned to them and twenty-three air squadrons with seven types of aircraft. The scenario begins with the Iraqi army occupying Kuwait. The primary measure of effectiveness (MOE) used in evaluating the ground war is command strength.

The unfortunate part of the unclassified scenario is the blue side heavily overpowers red. In an attempt to keep the red aircraft flying for the baseline scenario, we deleted ten squadrons of blue aircraft plus one carrier battle group. Still the red aircraft flew for only a few days. Of the nine Iraqi commands, four of them are dummy

commands we turned into armor or mechanized units to generate targets as a sink for blue aircraft but play no role in the metamodel. In order to put teeth into the ground troops, changes in were also made to the unit types. The identification of the squadrons deleted and the changes to the ground commands are summarized in Appendix C. All of this made for a stable baseline ground war involving the four commands shown in Figure 4.

4.3. Estimating the Leontief Model

In order to use the Leontief Model as a metamodel, data from THUNDER must be passed and read back into the simulation at regular intervals. A special version of the THUNDER executable was developed for use in this thesis. This version requires that the ground combat cycle and the red air cycle be set for identical periods. The period was set for twelve hours for our research. At the end of each cycle the simulation is interrupted and an output file called SIMU90 is copied into the Reports directory. An example of selected pages of the report may be found in Appendix C.

Once the SIMU90 file has been copied to the Reports folder, the file tt2leon is called by THUNDER to pass control to the metamodel. The tt2leon file must process the output information and copy a file named SIMU91 into the Reports folder. The SIMU91 file requires unit number, and a percentage between zero and one, where one is no degrade to the unit. It also requires for each squadron and mission type the squadron number, mission type, and the planned number of sorties.

The unit degrade percentage reduces the strength of the unit which is then used to determine its posture and FLOT advantage as discussed earlier. The number of sorties planned for each mission type for each squadron is then used in allocating aircraft to

targets using the methodology described earlier. The number of sorties passed back to THUNDER using the SIMU91 file must be less than or equal to the original number of sorties in the output file.

The files used to collect the data for estimation, generating a linear program for CPLEX, and the final tt2leon file used as an interface may be found in Appendix D. All of the files except for the CPLEX solver are written in Perl. I used a CPLEX example for the solver program that was modified to print the solution to a text file. I had to write programs that converted the data into linear programs for CPLEX to read because we did not have a program such as GAMS or MPL available for our Unix machines. Should anyone attempt to replicate this method with a full-scale database, I highly recommend using one of these programs to speed the data conversion and solutions to the linear program. Our problem was small and the interface only takes about fourteen seconds to read the problem, solve the linear program and return control back to THUNDER.

The ground data collection file reads the strength of each unit and aggregates that strength to the command level. For this entire application, the sector consumption variable is the amount of strength forces "exported" directly to the blue side. Therefore, each unit that is on-line is then considered to be consumption and is aggregated to the command level for an estimated value of y . The formulas for this aggregation are

$$TO_c = \frac{\sum US}{\#Units} \times 100 \text{ and } y_c = \frac{\sum SOL}{\#Units} \times 100, \quad (4-1)$$

where TO_c is the Total Output for a command, US is the unit strength passed from the SIMU90 file, and SOL is the strength of on-line units. Each of the unit strengths is a percentage and if all units are at 100 percent or 1.0, then TO_c is equal to 100. The

formula for consumption has a unique aspect to it in that if two units are at .89 of original unit strength then the command strength is also .89, but if one unit is on-line the consumption for the command is .445 not .5. This was done purposely for both the estimation and use of the Leontief model since we use the consumption value to determine if a degrade will go into effect. In essence, this approach is very conservative in that it will only allow degrades when unit strength is so low that the online troops can no longer be supported.

Although the air data was not used for this scenario, the collection file summed the number of sorties flown in the last cycle as the Total Output for the squadron. The three general missions identified in Air Force Manual 1-1 as Aerospace Control, Force Application, and Force Support determined consumption for the air forces. Under the heading of Aerospace Control Missions, the offensive counter air (OCA) mission is defined as, "Missions that take the initiative to destroy the enemy's ability to operate in the aerospace environment by attacking systems (or their support systems)"(AFM 1-1, pg 104). In THUNDER, three missions are defined by their offensive nature and fit our definition of "exporting" force directly. They are the OCA, strategic target interdiction (STI), and interdiction (INT) missions. These missions were then summed and used as the consumption value for y_3 in the estimation of the Leontief model. The interface for simulation runs in which the metamodel is active collects the planned sorties and uses them in the same way.

This selection of ground unit strength and planned sorties as variables uses a valuable aspect of THUNDER because availability of resources is factored into them. The strength of the ground unit is determined by the amount of equipment and resources

available, as is the number of sorties planned and flown. The approach is valuable for three reasons. First, we have eliminated the time needed to calculate the primary resources matrix. Second, the limits on total output discussed in Figure 2 are built into the constraints passed by THUNDER. Finally, the Leontief homogeneity of resources assumption is bypassed as THUNDER meticulously tracks resources through the logistics network. The code in Appendix D has the section for the primary resources commented out with the number symbol (#) should any one wish to consider resources as constraints without modifying those in the THUNDER database.

4.4. The Leontief Metamodel

The Leontief metamodel is estimated using the methodology in Chapter 3 and the data from the four commands displayed in Figure 4. Trying to involve the air squadrons in this demonstration was impossible due to the poor structure of the unclassified database. Since the aircraft would not fly during the entire ten-day war, their input to the ground sector was zero—a result inherent in the estimation procedure. Consider the following two equations from an arbitrary data set.

$$\begin{array}{rcl} 100 - a_{11}100 - a_{12}20 - e_1 & = & 35 \\ 0 - a_{11}0 - a_{12}10 - e_2 & = & 0 \end{array}$$

The first equation states the total output of sector one is 100 while the second equation states sector one's total output is zero. When minimizing the excess variables, a_{12} must equal zero in the second equation. This makes sense from Leontief's point of view since sector two has positive output with no inputs from sector one in the second equation. Because the squadrons only flew for the first few days, equations similar to the second one appeared over and over resulting in the aircraft having no input to the ground units.

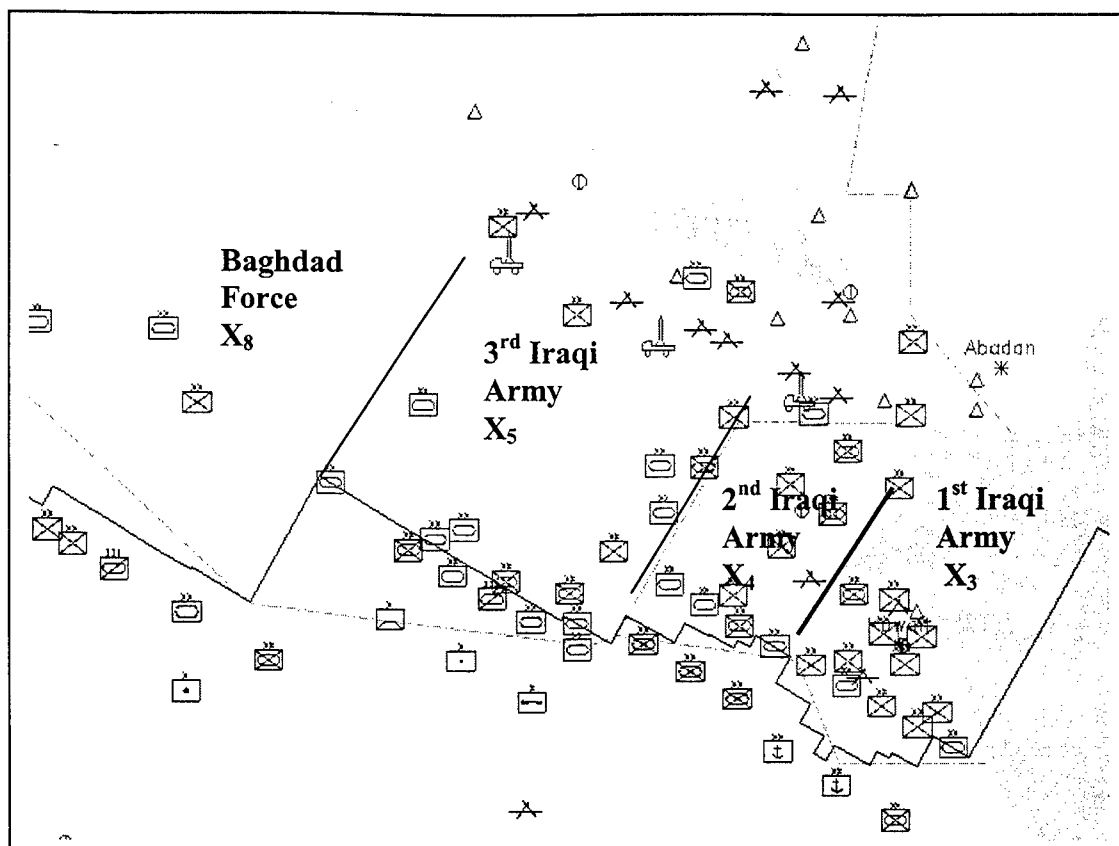


Figure 4. ME Scenario Situation Map

Table 14 shows the estimations for the ground commands with no bounds on any of the variables. There are two important structural items to notice about this table. The first is that the 3rd Iraqi Army's strength, X_5 , appears to be tied heavily to that of the 2nd Iraqi Army, X_4 .

Table 14. Originally Estimated Ground Commands

Commands	X_3	X_4	X_5	X_8
X_3	0.48	0	0	0
X_4	0	0.53	0	0.018
X_5	0	0.40	0.04	0.08
X_8	0	0	0	0

Also X_8 , the Baghdad Force, has zeroes across the entire row. This is because the Baghdad Force had only five units and during some portions of the war all of the units

were on-line. When the total output is equal to the total on-line, then all of the input coefficients will be zero as discussed with the squadron aircraft a moment ago. The power of this estimation is 0.85 (1-9491/63800), which means the interactions in the Leontief Input-Output model captures 85% of the variation between total output of each sector and the resulting attack strength.

We are now in a position to manipulate the input coefficient matrix into something that makes sense for our battle. THUNDER is somewhat stove-piped so we incorporate minimal interactions by assigning a rule that each command would support the command on its flank with an input coefficient of 0.05 with the exception of the Baghdad command, which can only be supported since all of its units are on-line through much of the battle. After assigning bounds to force the interaction terms to be greater than or equal to 0.05, the linear program is resolved to provide a new solution. Table 15 contains the input coefficient matrix modeling the ground commands for the rest of this exercise. In order to avoid making software modifications, the full 32 x 32 matrix with all of the squadrons and commands was used with zeros in their place. The power of the new estimation is still 0.85 percent meaning the sum of errors increased only slightly due to the forced interaction. We have tried to make our calculations as conservative as possible at every step so these values will never underestimate consumption.

Table 15. Input Coefficient Matrix Used

Commands	X ₃	X ₄	X ₅	X ₈
X ₃	0.42	0.05	0	0
X ₄	0.05	0.45	0.05	0
X ₅	0	0.40	0.04	0.08
X ₈	0	0	0	0

Now that we had interaction between the ground commands, we only needed strategic targets. The two selected for this project were a communications center and a command bunker. Arbitrarily we assigned the communications center to only be needed by 2nd Iraqi Army, X_4 , while the command bunker is required by all four commands. Each target was assigned a value of how much that target contributed to all communications or command, respectively. It was decided that the communications center would account for two-thirds of all communications for X_4 . The strategic targets in these examples are either fully operational or destroyed. When fully operational, communications totaled 150 and when destroyed the value was reduced to 50. Similarly, the command bunker accounted for about 60% of all higher command for the four armies. When the bunker was operational, command was 170 and when destroyed totaled 70. The units of these sectors are unimportant as the example is simply notional.

With the targets and their relative value in place, the constraints to the commands can be added to the linear program. The constraints added are

$$\begin{aligned} X_{33} - .65 X_4 &\geq 0 \text{ Communications} \\ X_{34} - .27 X_3 - .27 X_4 - .27 X_5 - .1 X_8 &\geq 0 \text{ Command} \end{aligned}$$

The communications constraint implies that when 2nd Iraqi Army, X_4 , is at 100 percent it requires 65 units of communication. Notice that when the communications bunker is destroyed only 50 units of communication are available and we would expect a direct effect degrade to X_4 . On the other hand if the strength of X_4 is less than 76 percent then there will be no direct effect on strength as a result of losing communications since only 49 units are required. Similarly, the command constraint maintains that if all units are at 100 percent the armies require 91 units of higher command orders.

A degrade in strength due to destruction of selected targets does not necessarily force degrades being passed back to THUNDER. Degrades are only passed back when consumption may not be maintained. In other words, if there is enough slack in the system that a command strength may be reduced but still have enough resources to maintain their original attack rate, a degrade will not be passed back to THUNDER. This is made more likely by the way we computed the consumption rates as mentioned earlier. Hence, the metamodel allows resources to be reallocated, if appropriate.

4.5. Results of Strategic Effects Through Leontief Metamodel

To demonstrate the mathematics behind the degrades passed to THUNDER as a result of strategic target destruction, one replication of the ten-day war will be used. The objective function is to maximize the consumption values for all sectors bounded by the original values read in for both total output and consumption from the SIMU90 file. The relevant constraints for the problem are those that deal with the four ground commands of interest:

$$\begin{array}{rcl}
 0.58 X_3 - 0.05 X_4 - y_3 & \geq & 0 & \text{X3} \\
 - 0.05 X_3 + 0.55 X_4 - 0.05 X_5 - y_4 & \geq & 0 & \text{X4} \\
 - 0.40 X_4 + 0.96 X_5 - 0.08 X_8 - y_5 & \geq & 0 & \text{X5} \\
 X_{33} - 0.65 X_4 & \geq & 0 & \text{Comm} \\
 X_{34} - 0.27 X_3 - 0.27 X_4 - 0.27 X_5 - 0.10 X_8 & \geq & 0 & \text{Cmnd}
 \end{array}$$

The first constraint limits the amount of support the 1st Iraqi Army, X_3 , may provide for the 2nd Iraqi Army, X_4 . Similarly, the second constraint distributes the total output available to the armies on each flank of 2nd Iraqi Army, X_4 . The third constraint distributes the strength of the 3rd Iraqi Army and the last two are the constraints for the communications center and command bunker, respectively.

The first scenario is with the communications center being destroyed in the first twelve hours on the second day (simulation time 2.5) of the war. The bottom scale of Figure 5 begins with the first half day of the war already adjudicated and the attrition rates are those normally generated in THUNDER. The charts in Figure 5 only depict one replication and are for a reference of the size and duration of degrades returned to THUNDER.

In this scenario **only the 2nd Iraqi Army, X_4 , is directly affected by the strike.** The heavy line on the charts shows the strengths at each input from THUNDER with the Leontief metamodel in place. The dotted line is the strength THUNDER reports with no strategic effects in place. At simulation time 2.5, the two lines are the same and the metamodel captures the destruction of the communications center. The Comm constraint is binding on X_4 limiting that commands total output to 76.92. This value is in turn binding on X_3 at 73.25 and on X_5 at 69.88 due to constraint **X4**. The value of X_8 is bound at 73.16 by the value of X_5 in the **X5** constraint. The value of y_4 is maximized at 35.15, missing the original consumption value from THUNDER of 36.35 and triggering a degrade. The degrades returned to THUNDER are the new values of total output divided by the original total output value passed in from THUNDER. In this instance, the degrades passed to THUNDER are 0.77 for X_3 (73.25/95.67), 0.81 for X_4 (76.92/94.69), 0.74 for X_5 (69.88/94.74) and 0.78 for X_8 (73.16/93.16).

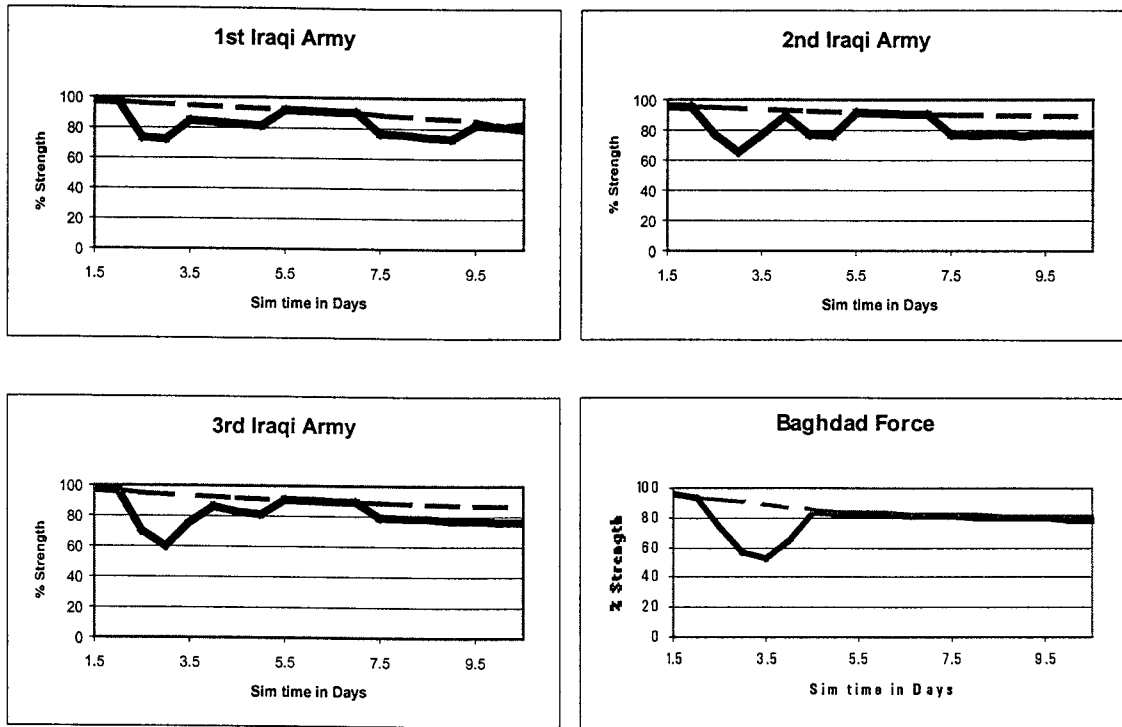


Figure 5. Effects from Destruction of Communications Center

The degrades are then used by THUNDER to adjudicate the battle and the numbers at the next cycle, day 3, are output and the returned values are 72.29 for X_3 , 65.07 for X_4 , 68.53 for X_5 , and 72.36 for X_8 . The Leontief model still has not satisfied all of its consumption constraints, and a second round of degrades is initiated. On day 3.5 the consumption constraints are nearly satisfied and the degrades begin to ease off. Notice that the strength of X_4 is reported above 76.92. This is because the other armies are also weaker and do not demand as much support. The Leontief model allows for a realistic reallocation of resources. As a result there is enough slack in the system that no further degrades are imposed by the metamodel and strengths are again calculated based on the resources available in THUNDER.

Several points need to be made about the communications center. First, although the attack only affects the 2nd Army directly, cascading effects are seen in the other armies. Also, as was pointed out in equation (4- 1), when the command strengths diminish the consumption variable becomes smaller at a faster rate because they are based on the maximum possible strength. A third point is that if the only consumption value shorted is associated with constraint **X4**, why did we not just degrade **X4**? The answer to that lies in the mathematics of the metamodel. If we only degraded unit **X4**, the consumption value would never be met since constraint **X4** requires a multiplier of **X3** and **X5** be taken from the net value of the original total output. When consumption cannot be met, the whole system is broken. It is normally only one tight constraint that will always be the limiting factor in a network styled model. Finally, the communications bunker is never restored yet the degrades taper off because each command loses strength allowing the consumption variables to be met.

The next strategic target to look at is the command bunker. In a scenario similar to that above, the command bunker is destroyed on day 1.5 and the **Cmnd** constraint is binding. The results are illustrated for each command in Figure 6.

In this scenario the degrade does not take effect immediately. The lag between target destruction and any noticeable impact is one day. On day 2.5 the **Cmnd** constraint becomes binding but because the consumption variables are small (the armies don't have a lot of units on-line yet) and can be met with the constrained total outputs for each command. Only on day 3.5, when the 1st and 3rd Armies commit more units to the battle, does the **X4** consumption variable again fail to be met and a degrade is imposed. In this case the degrades are shallower but have more endurance.

Now that we have shown two types of degrades, it is time to develop an experiment to determine which target set is best—Comm, Command, or both.

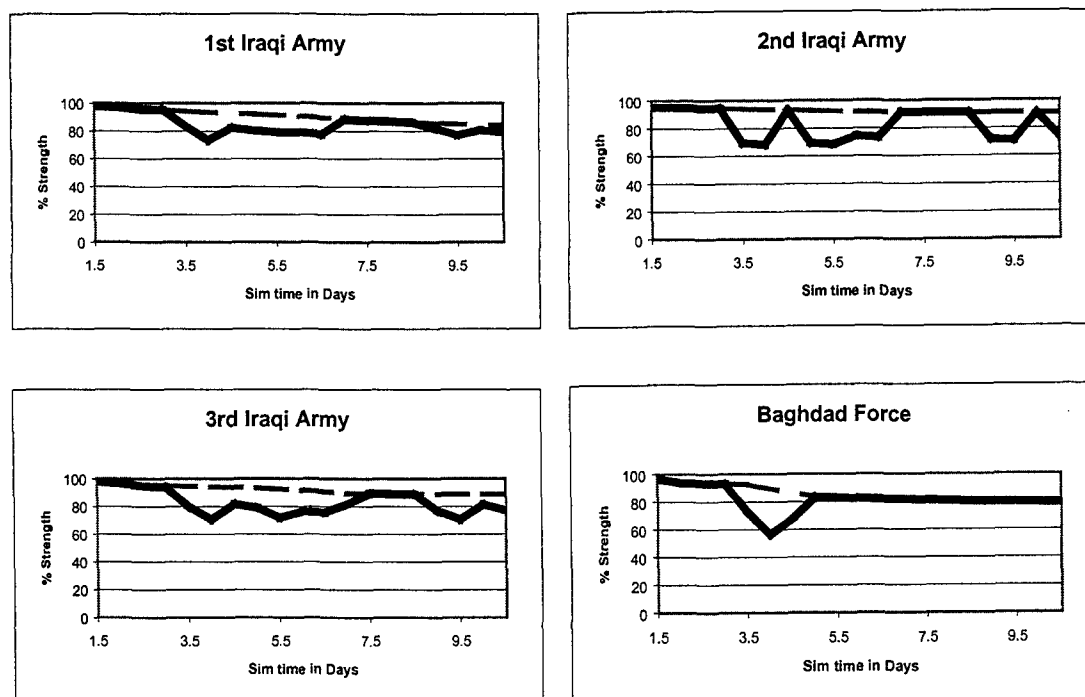


Figure 6. Degrades due to Command Bunker Destruction

4.6. An Experiment to Aid Policy Decisions

The time and size of initial degrade data was collected by randomly assigning an attack time to 30 runs of communications target only, an attack on a command bunker only, and an attack on both entities at the same time. The degrade time was randomly picked from a $U[2, 8]$ distribution where two and eight are the simulation time in days. The intent of the random draw was to reflect the stochastic environment of THUNDER. The uncertainty not only models the inability to predict exactly when the target will be attacked, but whether the strike was effective. The metamodel as explained earlier sometimes introduces a lag time between target destruction and the first degrade. The actual degrade time represents when the effect of target destruction is realized.

The time and the size of the degrade is calculated in the same manner as described in the earlier section. A data file containing each army's relative strength was output for each unit of time in the scenario. The time of target destruction, the actual time the degrade happened, and the amount of the initial degrade for each of the four army groups was extracted from the file. The four degrades were averaged to provide an overall degrade estimate. The times of the degrades for the 30 runs were sorted in chronological order. The results of the Command Bunker attack have at least a one-day lag in all cases.

A significant difference was found between the mean communications center degrade and mean command bunker degrade scenarios at a p-value of 0.024. A confidence interval was also constructed about the mean of each degrade for each scenario. This was done to compare the confidence intervals with a control variate variance reduction technique. The control variate technique applied uses the actual degrade time as the control variate using the regression equation

$$\text{Ave Degrade} = \beta_0 + \beta_1(\text{DegTime} - \text{AveDegTime}).$$

The regression equation was very significant according to the F-statistic in all three scenarios. The slope of β_1 was between -0.02 and -0.03 for all three scenarios. This suggests that for each day the attack is delayed the initial impact of the degrade will be 3% less on average.

Figure 7 illustrates the results of the control variate regression technique based on destroying the communications target uniformly between day two and day eight. From this illustration it is clear that the earlier the target is destroyed the larger the degrade to the units will be.

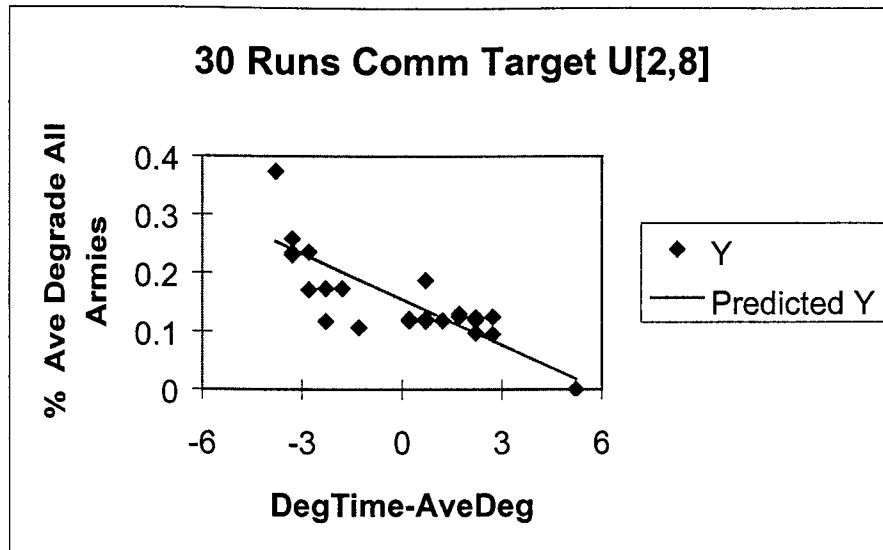


Figure 7. Use of a Control Variate

Figure 8 shows the aggregated results of all three target sets. The communications target dominates the parallel attack option. There are also two other things to point out. First, we get a definite non-linear output when only one of the commands is involved in the target set. In the case of the communications target, only the 2nd Iraqi Army was directly affected by the target yet the degrade across time appears to have an exponential decay to its average initial degrade across time. The other thing to point out is that as more commands are involved in the target set, the impact of the target's destruction becomes more linear although the duration of the degrades seems to last longer.

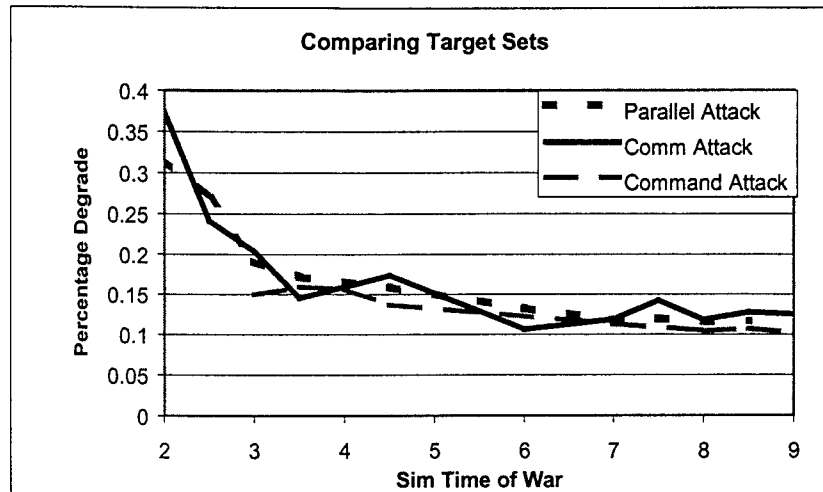


Figure 8. Comparing Initial Degrades Against All Target Sets

We can conclude from this data that an attack on the communications bunker is preferable to that of the command bunker as illustrated in Table 16. The same conclusion cannot be made on the difference between the communications bunker and the parallel attack. Here we can support the proposal that if concentrating our efforts only on the communications center can improve the probability of destroying the target earlier then a larger degrade will be realized.

Table 16. Comparison of Confidence Interval (CI) vs Control Variate (CV)

$\alpha=.10$	Comm Center			Cmnd Bunker			Parallel Attack		
	Lower Limit	Mean	Upper Limit	Lower Limit	Mean	Upper Limit	Lower Limit	Mean	Upper Limit
CI	13.0	15.5	17.9	9.3	11.2	13.2	11.9	14.0	16.1
CV	14.0	15.5	16.9	10.4	11.2	12.0	12.9	14.0	15.0

5. Conclusions and Follow-on Research

A variety of military strategic effects have been demonstrated as a result of using the Leontief Input-Output model as a metamodel for THUNDER. We have demonstrated a system that captures direct, indirect, immediate, and delayed effects due to the attack of a strategic target. Further, we have demonstrated a methodology that uses this information to prioritize strategic targets for desired results.

The target set used for our demonstration consisted of only two independent targets. As the target sets get larger and more interdependent, the complexity and simplicity of this system will realize its full potential.

We have demonstrated that the metamodel may be estimated with a great deal of accuracy using the data from a combat model. The metamodel enables us to incorporate un-modeled interactions into current combat models. We would like to see this approach tried to adjust for historical impacts of an attack.

This research focused only on military strategic effects but has the potential for measuring economic and social effects as well. National economic input-output tables already exist for over 60 nations and the quality of the tables is well documented in economic journals. Sociologists are also beginning to use the Leontief Input-Output model to gain a macro perspective on societal issues (Namboodiri, 1993). If the target sets in our combat models have an impact in these other models, the DoD will have an analytical tool for estimating what has been considered up to this point qualitative economic and sociological impacts of its strategies. Hence, the three rings of Clausewitz shown in Figure 1 could be modeled.

The Leontief model also has a pricing mechanism that forms the dual in an economic model. In economics the dual simply states that we attempt to maximize production while minimizing cost. Future research should be directed toward using a risk index for targets to form the dual of the target set in which we try to maximize the size of degrades while minimizing the risk of attacking targets.

Logisticians should seriously consider using the dynamic Leontief model to improve their systems. In the past logistics has been assumed away in war game and command and control exercises because there was no method to include them. By using the dynamic model for logistics, the matrix of capital goods or stock becomes the resources available for the commanders playing in the combat model. In other words, the capital stock matrix in the logistics game is the primary resource restrictions in the combat game. If there is an error at the input of the logistics game, the combat commanders must live with it until it can be rectified through the game scenario. The benefit of this type of war game exercise is the ability to evaluate hundreds of strategies in a short period of time.

Finally, strategic analysts should consider this approach. Complex attacks such as information warfare or nuclear weapon attacks that simultaneously degrade many sectors could be modeled using this approach. It is very clear that the direction of the community has moved toward larger and larger models with increasing complexity and detail. Unfortunately, as the models grow larger and more complex it is more difficult to distinguish what assumptions are driving the outcome of the model and testing a large multiple of strategies is out of the question. The methods we have described here accomplish two things. First, the interrelationships of a complex model are highlighted

in a simple fashion. Second, the approach enables the analyst to perform sensitivity analysis on all of the parameters and test multiple strategies in a fraction of the time. This **approach will make the analyst more relevant** when the important decisions on the size and structure of our military force are made.

Bibliography

- Air Force Doctrine Center (AFDC). Strategic and Indirect Effects: Defining and Modeling. Unpublished briefing slides available at <http://www.hqafdc.maxwell.af.mil/>. Downloaded 31 March 1999.
- Clausewitz, Carl von. On War. Edited and translated by Michael Howard and Peter Peret. Princeton, N.J.: Princeton University Press, 1976.
- Department of the Air Force. Air Warfare. AFDD 2-1. (Draft) Washington: HQ, USAF, (downloaded from AFDC home page) December 1999.
- Department of the Air Force. Basic Aerospace Doctrine of the USAF. Air Force Manual 1-1, Volume II. Washington: HQ, USAF, March 1992.
- Dorfman, Robert and others. Linear Programming and Economic Analysis. New York: Dover Publications, Inc., 1987.
- Landefeld, J. Steven and McCulla, Stephanie H. Wassily Leontief and His Contributions to Economic Accounting. *Survey of Current Business*. Bureau of Economic Analysis. March 1999.
(<http://www.bea.doc.gov/bea/ARTICLES/NATIONAL/Inputout/1999/0399leon.pdf>)
- Leontief, Wassily. Input-Output Economics. New York: Oxford University Press, 1986.
- Meilinger, Phillip S. 10 Propositions Regarding Air Power. Air Force History and Museums Program: GPO, 1995.
- Mets, David R. The Air Campaign: John Warden and the Classical Airpower Theorists. Air University Press, Maxwell AFB, Alabama, 1998.
- Miller, Ronald E. and Blair, Peter D. Input-Output Analysis: Foundations and Extensions. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- Namboodiri, K. and Corwin, R.G., The Logic and Method of Macrosociology, Praeger Publishers, Westport CT 06881, 1993.
- Warden, John A. III. (1989) The Air Campaign: Planning for Combat. Washington: Brassey's, 1989. (First published by NDU Press, 1988).
- Warden, John A. III. (1995) "The Enemy as a System," Airpower Journal. Vol IX, No.1, Spring 1995, pp. 40-55.

Appendix A: Leontief Input-Output Data Generation Program

Visual Basic Program to Generate Model Data

```
Sub GenNumbers()
```

```
'This program was written on October 21, 1999
```

```
'Generates data points for 100 iterations of a Leontief Input-Output Model
```

```
Randomize
```

```
Dim Cons(4) As Integer
```

```
Dim OutX(10) As Double
```

```
For i = 1 To 100
```

```
'This section generates random consumption values.
```

```
'The first generates a random number between 50 and 100, the second between 60 and 120 etc
```

```
Cons(1) = Int((50 * Rnd) + 50)
```

```
Cons(2) = Int((60 * Rnd) + 60)
```

```
Cons(3) = 0
```

```
Cons(4) = Int((60 * Rnd) + 10)
```

```
'This generates the total output based on consumption using equation (2-5)
```

```
'The program assumes that the A matrix is located in cells M4:P7
```

```
OutX(1) = Sheet2.Cells(4, 13).Value * Cons(1) + Sheet2.Cells(4, 14).Value * Cons(2) +  
Sheet2.Cells(4, 15).Value * Cons(3) + Sheet2.Cells(4, 16).Value * Cons(4)
```

```
OutX(2) = Sheet2.Cells(5, 13).Value * Cons(1) + Sheet2.Cells(5, 14).Value * Cons(2) +  
Sheet2.Cells(5, 15).Value * Cons(3) + Sheet2.Cells(5, 16).Value * Cons(4)
```

```
OutX(3) = Sheet2.Cells(6, 13).Value * Cons(1) + Sheet2.Cells(6, 14).Value * Cons(2) +  
Sheet2.Cells(6, 15).Value * Cons(3) + Sheet2.Cells(6, 16).Value * Cons(4)
```

```
OutX(4) = Sheet2.Cells(7, 13).Value * Cons(1) + Sheet2.Cells(7, 14).Value * Cons(2) +  
Sheet2.Cells(7, 15).Value * Cons(3) + Sheet2.Cells(7, 16).Value * Cons(4)
```

```
OutX(5) = Sheet2.Cells(13, 13).Value * Cons(1) + Sheet2.Cells(13, 14).Value * Cons(2) +  
Sheet2.Cells(13, 15).Value * Cons(3) + Sheet2.Cells(13, 16).Value * Cons(4)
```

```
OutX(6) = Sheet2.Cells(14, 13).Value * Cons(1) + Sheet2.Cells(14, 14).Value * Cons(2) +  
Sheet2.Cells(14, 15).Value * Cons(3) + Sheet2.Cells(14, 16).Value * Cons(4)
```

```
OutX(7) = Sheet2.Cells(15, 13).Value * Cons(1) + Sheet2.Cells(15, 14).Value * Cons(2) +  
Sheet2.Cells(15, 15).Value * Cons(3) + Sheet2.Cells(15, 16).Value * Cons(4)
```

```
OutX(8) = Sheet2.Cells(16, 13).Value * Cons(1) + Sheet2.Cells(16, 14).Value * Cons(2) +  
Sheet2.Cells(16, 15).Value * Cons(3) + Sheet2.Cells(16, 16).Value * Cons(4)
```

```
OutX(9) = Sheet2.Cells(17, 13).Value * Cons(1) + Sheet2.Cells(17, 14).Value * Cons(2) +  
Sheet2.Cells(17, 15).Value * Cons(3) + Sheet2.Cells(17, 16).Value * Cons(4)
```

```
OutX(10) = Sheet2.Cells(18, 13).Value * Cons(1) + Sheet2.Cells(18, 14).Value * Cons(2) +  
Sheet2.Cells(18, 15).Value * Cons(3) + Sheet2.Cells(18, 16).Value * Cons(4)
```

```
'This outputs the values generated at each iteration columns in Sheet 2
```

```
Sheet3.Cells(1 + i, 1).Value = OutX(1)
Sheet3.Cells(1 + i, 2).Value = OutX(2)
Sheet3.Cells(1 + i, 3).Value = OutX(3)
Sheet3.Cells(1 + i, 4).Value = OutX(4)
Sheet3.Cells(1 + i, 5).Value = OutX(5)
Sheet3.Cells(1 + i, 6).Value = OutX(6)
Sheet3.Cells(1 + i, 7).Value = OutX(7)
Sheet3.Cells(1 + i, 8).Value = OutX(8)
Sheet3.Cells(1 + i, 9).Value = OutX(9)
Sheet3.Cells(1 + i, 10).Value = OutX(10)
```

'This prints the randomly generated consumption values to Sheet 2

```
Sheet3.Cells(1 + i, 14).Value = Cons(1)
Sheet3.Cells(1 + i, 15).Value = Cons(2)
Sheet3.Cells(1 + i, 16).Value = Cons(3)
Sheet3.Cells(1 + i, 17).Value = Cons(4)
```

Next i

End Sub

First 30 Data Points Generated

X1	X2	X3	X4	P1	P2	P3	P4	P5	P6	C1	C2	C3	C4
115	258	100	62	645	323	31	347	1473	587	80	117	0	62
113	211	87	56	565	283	27	314	1315	518	82	85	0	56
120	166	76	24	505	253	24	134	756	384	92	106	0	24
72	123	52	27	340	170	16	151	683	291	53	60	0	27
131	223	95	62	619	309	29	347	1450	569	97	83	0	62
103	211	84	59	545	272	26	330	1349	515	73	79	0	59
85	181	71	30	461	231	22	168	823	376	60	109	0	30
115	209	87	64	566	283	27	358	1443	543	84	67	0	64
126	157	75	30	502	251	23	168	848	400	98	85	0	30
89	180	72	52	466	233	22	291	1178	445	63	64	0	52
125	220	92	49	604	302	29	274	1231	521	92	107	0	49
122	145	71	16	474	237	22	90	603	340	95	102	0	16
85	140	60	16	393	197	19	90	547	291	63	98	0	16
104	222	87	51	564	282	27	286	1236	503	73	105	0	51
110	153	70	20	464	232	22	112	662	347	84	102	0	20
82	166	66	42	430	215	21	235	991	393	58	71	0	42
74	169	65	21	420	210	20	118	649	324	51	116	0	21
127	150	74	19	493	246	23	106	665	361	99	101	0	19
78	120	53	18	348	174	16	101	545	269	59	75	0	18
86	181	72	53	464	232	22	297	1193	447	61	63	0	53
125	202	87	36	573	287	27	202	999	463	93	116	0	36
84	206	78	63	501	250	24	353	1382	500	57	67	0	63
112	146	69	31	455	228	21	174	831	374	86	73	0	31
71	135	55	32	358	179	17	179	778	318	51	62	0	32
128	205	89	47	583	292	28	263	1183	502	95	97	0	47
115	203	85	35	555	277	26	196	971	448	84	119	0	35
96	213	83	60	535	268	26	336	1358	512	67	79	0	60
134	234	98	59	642	321	31	330	1419	575	98	100	0	59
87	172	69	42	449	224	22	235	1005	404	62	76	0	42
115	258	100	62	645	323	31	347	1473	587	80	117	0	62
113	211	87	56	565	283	27	314	1315	518	82	85	0	56
120	166	76	24	505	253	24	134	756	384	92	106	0	24
72	123	52	27	340	170	16	151	683	291	53	60	0	27
131	223	95	62	619	309	29	347	1450	569	97	83	0	62
103	211	84	59	545	272	26	330	1349	515	73	79	0	59
85	181	71	30	461	231	22	168	823	376	60	109	0	30
115	209	87	64	566	283	27	358	1443	543	84	67	0	64
126	157	75	30	502	251	23	168	848	400	98	85	0	30
89	180	72	52	466	233	22	291	1178	445	63	64	0	52
125	220	92	49	604	302	29	274	1231	521	92	107	0	49
122	145	71	16	474	237	22	90	603	340	95	102	0	16
85	140	60	16	393	197	19	90	547	291	63	98	0	16
104	222	87	51	564	282	27	286	1236	503	73	105	0	51
110	153	70	20	464	232	22	112	662	347	84	102	0	20
82	166	66	42	430	215	21	235	991	393	58	71	0	42

Appendix B. Lingo Formats for Estimating Input Coefficients

Minimized the Excess

MODEL:

SETS:

RUNS/1..100/;

DATA/1..4/;

SECTORS(RUNS,DATA):OUTPUTS,C,E;

ARCS(DATA,DATA)/1,1 1,2 1,3 1,4 2,1 2,2 2,3 2,4 3,1 3,2 3,3 3,4 4,1 4,2 4,3 4,4/:AIJ;

ENDSETS

Min=@SUM(SECTORS:E);

@FOR(SECTORS(N,J):OUTPUTS(N,J)-@SUM(ARCS(I,J):AIJ(J,I)*OUTPUTS(N,I))-
C(N,J)-E(N,J)= 0);

@FOR(ARCS:AIJ>=0);

@FOR(SECTORS:E>=0);

DATA:

OUTPUTS=@FILE(TOTALOUT.csv);

C=@FILE(cons.csv);

ENDDATA

END

Arc 3,3 is removed to for
proper estimation.

Minimizing the Excess for Primary Resources

MODEL:

SETS:

RUNS/1..100/;

DATA/1..4/;

RES/1..6/;

SECTORS(RUNS,DATA):OUTPUTS;

PRIMINPT(RUNS,RES):TOTPRIM,E;

ARCS(RES,DATA)/1,1 1,2 1,3 1,4 2,1 2,2 2,3 2,4 3,1 3,2 3,3 3,4 4,1 4,2 4,3 4,4
5,1 5,2 5,3 5,4 6,1 6,2 6,3 6,4/:PIJ;

ENDSETS

Min=@SUM(PRIMINPT:E);

@FOR(PRIMINPT(N,K):@SUM(SECTORS(N,J): OUTPUTS(N,J) * PIJ(K,J))+
E(N,K)-TOTPRIM(N,K)=0);

@FOR(ARCS:PIJ>=0);

@FOR(PRIMINPT:E>=0);

DATA:

OUTPUTS=@FILE(TOTALOUT.csv);

TOTPRIM=@FILE(PRIMRES.csv);

ENDDATA

END

Appendix C. Changes Made in THUNDER Databases

Table 17. Blue Aircraft Squadrons Deleted

11102	11502
11103	11503
11104	11504
11105	11506
11501	11601
USN Carrier Group 1 Auth Aircraft set to zero	

Table 18. Configuration of Red Commands

Commands	Original Unit Configuration	Modeled with Teeth
2009	2 Dummy units	2 Infantry (Inf)
2010	2 Dummy	2 Inf
2011	12 Inf	9 Inf, 2 Armor, 1 Mech
2012	8 Inf, 1 Armor, 3 Mech	6 Inf, 3 Armor, 3 Mech
2013	11 Inf, 2 Armor, 1 Mech	3 Inf, 7 Armor, 4 Mech
2014	2 Dummy	1 Armor, 1 Mech
2015	1 Dummy	1 Inf
2016	3 Inf, 2 Dummy	3 Armor, 2 Inf
2017	1 Dummy	1 Mech

SIMU90 File Extracts

Extract for the strategic targets

TIME 1.0000
 STR 20010 "Bag Power Plant" 1.000
 STR 20020 "Bag Refinery" 1.000
 STR 20030 "Bag Main Bunker" 1.000
 STR 20040 "Bag Comm Center" 1.000
 STR 20120 "Comm Center" 1.000
 STR 20050 "Bag Scndry Bunker" 1.000
 STR 20130 "Cmd Bunker" 1.000

Extract from the unit

UNIT 2101 "1 1 ID" 2011 0. 0. "ON LINE" BADD
 UNAMMO 2101 0 7260 0
 UNBULK 2101 0 4540 0
 UNPOL 2101 0 2720 0
 UNWATER 2101 0 4540 0
 UNEQ 2101 2100 8 "RED TANK" 0 325 0
 UNEQ 2101 2200 9 "RED APC" 0 250 0
 UNEQ 2101 2300 10 "RED HELO" 0 12 0
 UNEQ 2101 2400 11 "RED HEAVY ARTY" 0 25 0
 UNEQ 2101 2500 12 "RED LIGHT ARTY" 0 125 0
 UNEQ 2101 2600 13 "RED INFANTRY" 0 33 0
 UNEQ 2101 2700 14 "RED AD GUN" 0 16 0
 UNEQ 2101 2710 15 "RED SHOULDER-FIRED SAM" 0 12 0
 UNEQ 2101 2720 16 "RED MOBILE SAM" 0 8 0

Extract from the squadron

SQN 22502 "SU25 2 " 2005
 SQAMMO 22502 0 869 0
 SQPOL 22502 0 27216 27216
 SQSORT 22502 DCA 0 0
 SQSORT 22502 ODCA 0 0
 SQSORT 22502 HVAA 0 0
 SQSORT 22502 BARCAP 0 0
 SQSORT 22502 FSWP 0 0
 SQSORT 22502 AIRESC 0 0
 SQSORT 22502 STI 0 0
 SQSORT 22502 CAS 0 0
 SQSORT 22502 BAI 0 0
 SQSORT 22502 INT 0 0
 SQSORT 22502 OCA 0 0
 SQSORT 22502 OTBM 0 0
 SQSORT 22502 DTBM 0 0
 SQSORT 22502 DSEAD 0 0
 SQSORT 22502 SSUP 0 0
 SQSORT 22502 CSUP 0 0
 SQSORT 22502 ESCSUP 0 0
 SQSORT 22502 SJAM 0 0
 SQSORT 22502 CJAM 0 0
 SQSORT 22502 ESCJAM 0 0
 SQSORT 22502 RECCE 0 0
 SQSORT 22502 SREC 0 0
 SQSORT 22502 AEW 0 0
 SQSORT 22502 AAR 0 0

SQSORT 22502 LIFT	0	0
SQSORT 22502 XXXX	0	0
SQSORT 22502 RESERVE	0	0
Extract from the squadron plan		
SQPLAN 22502 DCA	0	
SQPLAN 22502 ODCA	0	
SQPLAN 22502 HVAA	0	
SQPLAN 22502 BARCAP	0	
SQPLAN 22502 FSWP	0	
SQPLAN 22502 AIRESC	0	
SQPLAN 22502 STI	0	
SQPLAN 22502 CAS	70	
SQPLAN 22502 BAI	18	
SQPLAN 22502 INT	0	
SQPLAN 22502 OCA	0	
SQPLAN 22502 OTBM	0	
SQPLAN 22502 DTBM	0	
SQPLAN 22502 DSEAD	0	
SQPLAN 22502 SSUP	0	
SQPLAN 22502 CSUP	0	
SQPLAN 22502 ESCSUP	0	
SQPLAN 22502 SJAM	0	
SQPLAN 22502 CJAM	0	
SQPLAN 22502 ESCJAM	0	
SQPLAN 22502 RECCE	0	
SQPLAN 22502 SREC	0	
SQPLAN 22502 AEW	0	
SQPLAN 22502 AAR	0	
SQPLAN 22502 LIFT	0	
SQPLAN 22502 XXXX	0	
SQPLAN 22502 RESERVE	0	

Appendix D. Interface and Solver Files

tt2leon Collects Data

This Perl script collects data for squadrons and units, aggregates it to command level, and stores it in totalout, res and consump files for later use.

```
#!/usr/local/bin/perl

open(IN, "SIMU90");

$maxst{2009} = "2";
$maxst{2010} = "2";
$maxst{2011} = "12";
$maxst{2012} = "12";
$maxst{2013} = "14";
$maxst{2014} = "2";
$maxst{2015} = "1";
$maxst{2016} = "5";
$maxst{2017} = "1";

while (<IN>) {
    /\("[^"]+\")/ && ($x = $1) =~ s/ //g;
    s/\("[^"]+\")/$x/;

    ($f0,$f1,$f2,$f3,$f4,$f5,$f6,$f7) = split;

    if ($f0 =~ /^TIME/){
        chop($f1);chop($f1);chop($f1);
        $time = $f1;
    }

    elsif ($f0 =~ /^UNIT/){
        $strengths{$f3} += $f5;
        $cmdid += $f3;
        $unit{$f1} = $f5;
        $unitid = $f1;
    }

    elsif ($f0 =~ /^UNAMMO/){
        $res{$f0} += $f2;
        $resid += $f0;
    }

    elsif ($f0 =~ /^UNBULK/){
        $res{$f0} += $f2;
        $resid += $f0;
    }

    elsif ($f0 =~ /^UNPOL/){
```

```

        $res{$f0} += $f2;
        $resid += $f0;
    }

    elsif ($f0 =~ /^UNWATER/) {
        $res{$f0} += $f2;
        $resid += $f0;
    }

    elsif ($f0 =~ /^UNEQ/) {
        $res{$f2} += $f6;
        $resid += $f2;
    }

    elsif ($f0 =~ /^SQAMMO/) {
        $res{$f0} += $f2;
        $resid += $f0;
    }

    elsif ($f0 =~ /^SQPOL/) {
        $res{$f0} += $f2;
        $resid += $f0;
    }

    # this section grabs the Squadron IDs and total sorties
    elsif ($f0 =~ /^SQSORT/) {
        $strengths{$f1} += $f4;
        $cmdid += $f1;
    }

    if ($f6 =~ /^"ON/) {
        $cons{$f3} += $f5;
        $consid += $f3;
    }

    if ($f2 =~ /^STI/) {
        $cons{$f1} += $f4;
        $consid += $f1;
    }

    if ($f2 =~ /^OCA/) {
        $cons{$f1} += $f4;
        $consid += $f1;
    }

    if ($f2 =~ /^INT/) {
        $cons{$f1} += $f4;
        $consid += $f1;
    }

    if ($f0 =~ /^SQPLAN/) {
        $f10 = "$f1 + $f2";
        $plansort{$f10} = $f3;
        $mission{$f1} += $f2;
    }

```

```

        $sqnid = $f1;
        $sorties{$f2} = $f3;
    }

} # end of while loop

foreach $key (keys %maxst) {
    $strengths{$key} = ($strengths{$key} / $maxst{$key}) * 100;
    $strengths{$key} = substr($strengths{$key}, 0, 6);
}

foreach $key (keys %maxst) {
    $cons{$key} = ($cons{$key} / $maxst{$key}) * 100;
    $cons{$key} = substr($cons{$key}, 0, 6);
}

if ($time > 1.0) {

@heads = sort by_mostly_numeric keys %strengths;

# this section prints out the unit strengths
$outfile = "totalout";
if (-e $outfile) {
    open(OUT, ">>$outfile");
}
else {
    open(OUT, ">$outfile");
    print OUT "@heads\n";
}
for $key (@heads) {
    print OUT "$strengths{$key} ";
}
print OUT "\n";

# this section prints out the resources consumed
@resheads = sort by_mostly_numeric keys %res;

$outfile = "res";
if (-e $outfile) {
    open(OUT, ">>$outfile");
}
else {
    open(OUT, ">$outfile");
    print OUT "@resheads\n";
}
for $key (@resheads) {
    print OUT "$res{$key} ";
}

```

```

}
print OUT "\n";

# this section prints out the estimation for consumption

@consheads = sort by_mostly_numeric keys %cons;

$outfile = "consump";
if (-e $outfile) {
    open(OUT, ">>$outfile");
}
else {
    open(OUT, ">$outfile");
    print OUT "@consheads\n";
}
for $key (@consheads) {
    print OUT "$cons{$key} ";
}
print OUT "\n";
} # ends the if statement to not post data at time 1.000

# this is the beginning of the SIMU91 file

@unitheads = (2101..2112, 2201..2212, 2301..2309, 2313..2317, 2401..2403, 2034..2037, 2039, 2040, 2044,
2045, 2046, 2047);
#sort by_mostly_numeric keys %unit;

$outfile = "SIMU91";

open(OUT, ">$outfile");

for $key (@unitheads) {
    print OUT "UNIT $key 1.0\n";
}

@sqnheads = (22901, 22101..22106, 20101..20105, 22501, 22502, 22301..22304, 20000, 29902..29904, 29901);

#keys %mission;
@missheads = ("DCA", "ODCA", "HVAA", "BARCAP", "FSWP", "AIRES", "STI", "CAS", "BAI",
"INT", "OCA", "OTBM", "DTBM", "DSEAD", "SSUP", "CSUP", "ESCSUP", "SJAM", "CJAM",
"ESCJAM", "RECCE", "SREC", "AEW", "AAR", "LIFT", "XXXX", "RESERVE");

#keys %sorties;

for $key (@sqnheads) {
    for $key2 (@missheads) {
        $f11 = "$key + $key2";
        print OUT "SQN $key $key2 $plansort{$f11}\n";
    }
}

```

```
sub by_mostly_numeric {  
    ($a <=> $b) || ($a cmp $b);  
}  
  
rename("SIMU90", "SIMU90.$time");  
close (IN);  
close (OUT);
```

Makelp

This Perl script takes the totalout and consump files and turns them into an LP for CPLEX.

```
#!/usr/local/bin/perl

open(IN, "totalout");
open(CONS, "consump");

open(OUT, ">leontief.lp");

print OUT "minimize\n\n";

    for ($I = 1; $I <= 380; $I++) { #380 is the number of data points
        for ($J = 1; $J <= 32; $J++) { #32 is the number of sectors

            if ($I == 1 && $J == 1){
                print OUT "E($I,$J)";
            } # end if

            else {

                print OUT "+ E($I,$J)";
            } # end else
        } # end for J
        print OUT "\n";
    } # end for I

print OUT "\nst\n\n";

$z = 1; # this is used as a counter to increment the first index of E
while (<IN>){
    # reads in a line of total output data and splits the elements into an array

    @total = split;

    while (<CONS>) {

        # reads in a line of consumption data splits the elements into an array
        @cons = split;

        for ($j = 0; $j <= 31; $j++) { # 31 is the number of sectors -1 index starts at zero
            for ($k = 0; $k <= 31; $k++) {
                $token = 0;
                $l = $j + 1;
                $m = $k + 1;

                if ($j == 9) { # this takes out aii for A10,15-21 and 23-28
                    $token = 1;
                }
            }
        }
    }
}
```

```

    }
    elseif ($j >= 14 && $j <= 20) {
        $token = 1;
    }
    elseif ($j >= 22 && $j <= 27) {
        $token = 1;
    }
    if ($j == $k) {
        $token++;
    }

    if($token != 2){
        print OUT " - $total[$k]A($l,$m)";
    }

    } # end the for k loop
    $newtotal = $cons[$j]-$total[$j];

    print OUT " - E($z,$l) = $newtotal\n";

    }# end the for j loop
    $z++;
last;
} # end the data loop

```

```

} # end the last while loop

```

```

close (OUT);

```


CPLEX Solver

/*Modified by Tony Snodgrass 9 Feb 00

This file is an edited version of lpex2.c - Reading in and optimizing a problem to run this file follow the directions below. The modification made was to print the solution to a file called whatever the last argument you type into the argument list. For example: solver problem.lp p solution.txt calls the program solver reads in problem.lp using the primal simplex and writes the solution to a text file. The solution may be written in binary form using the .bin extension on the last argument.

*/

/* To run this example, command line arguments are required.

i.e., lpex2 filename method

where

filename is the name of the file, with .mps, .lp, or .sav extension

method is the optimization method

p or o primal simplex

d or t dual simplex

h barrier with crossover

b barrier without crossover

n network with dual simplex cleanup

Example:

lpex2 example.mps o

*/

/* Bring in the CPLEX function declarations and the C library

header file stdio.h with the following single include. */

#include "/usr/apps/cplex60/cplex.h"

/* Bring in the declarations for the string and character functions

and malloc */

#include <ctype.h>

#include <stdlib.h>

#include <string.h>

/* Include declarations for functions in this program */

#ifndef CPX_PROTOTYPE_MIN

static void

free_and_null (char **ptr),

usage (char *progname);

#else

static void

free_and_null (),

usage ();

#endif

```

#ifndef CPX_PROTOTYPE_MIN
int
main (int argc, char *argv[])
#else
int
main (argc, argv)
int  argc;
char *argv[];
#endif
{
    /* Declare and allocate space for the variables and arrays where we will
       store the optimization results including the status, objective value,
       variable values, dual values, row slacks and variable reduced costs. */

    int  solstat;
    double objval;
    double *x  = NULL;
    int  *cstat = NULL;
    int  *rstat = NULL;

    CPXENVptr  env = NULL;
    CPXLPptr   lp = NULL;
    int  status = 0;
    int  j;
    int  cur_numrows, cur_numcols;

    char  *basismsg;

    /* Check the command line arguments */

    if (( argc != 4 )
        || ( strchr ("podthbn", argv[2][0]) == NULL ) ) {
        usage (argv[0]);
        goto TERMINATE;
    }

    /* Initialize the CPLEX environment */

    env = CPXopenCPLEXdevelop (&status);

    /* If an error occurs, the status value indicates the reason for
       failure. A call to CPXgeterrorstring will produce the text of
       the error message. Note that CPXopenCPLEXdevelop produces no output,
       so the only way to see the cause of the error is to use
       CPXgeterrorstring. For other CPLEX routines, the errors will
       be seen if the CPX_PARAM_SCRIND indicator is set to CPX_ON. */

    if ( env == NULL ) {
        char  errmsg[1024];
        fprintf (stderr, "Could not open CPLEX environment.\n");
        CPXgeterrorstring (env, status, errmsg);
        fprintf (stderr, "%s", errmsg);
    }

```

```

    goto TERMINATE;
}

/* Turn off output to the screen */

status = CPXsetintparam (env, CPX_PARAM_SCRIND, CPX_OFF);
if ( status ) {
    fprintf (stderr,
        "Failure to turn on screen indicator, error %d.\n", status);
    goto TERMINATE;
}

/* Create the problem, using the filename as the problem name */

lp = CPXcreateprob (env, &status, argv[1]);

/* A returned pointer of NULL may mean that not enough memory
   was available or there was some other problem. In the case of
   failure, an error message will have been written to the error
   channel from inside CPLEX. In this example, the setting of
   the parameter CPX_PARAM_SCRIND causes the error message to
   appear on stdout. Note that most CPLEX routines return
   an error code to indicate the reason for failure. */

if ( lp == NULL ) {
    fprintf (stderr, "Failed to create LP.\n");
    goto TERMINATE;
}

/* Now read the file, and copy the data into the created lp */

status = CPXreadcopyprob (env, lp, argv[1], NULL);
if ( status ) {
    fprintf (stderr, "Failed to read and copy the problem data.\n");
    goto TERMINATE;
}

/* Optimize the problem and obtain solution. */

switch (argv[2][0]) {
    case 'o':
    case 'p':
        status = CPXprimopt (env, lp);
        break;
    case 'd':
    case 't':
        status = CPXdualopt (env, lp);
        break;
    case 'b':
        status = CPXbaropt (env, lp);
        break;
    case 'h':
        status = CPXhybbaropt (env, lp, 'p');
        break;
}

```

```

    case 'n':
        status = CPXhybnetopt (env, lp, 'd');
        break;
    default:
        status = -1;
        break;
}

if ( status ) {
    fprintf (stderr, "Failed to optimize LP.\n");
    goto TERMINATE;
}

solstat = CPXgetstat (env, lp);
status = CPXgetobjval (env, lp, &objval);

if ( status ) {
    fprintf (stderr, "Failed to obtain objective value.\n");
    goto TERMINATE;
}

printf ("Solution status %d. Objective value %.10g\n",
        solstat, objval);

/* The size of the problem should be obtained by asking CPLEX what
   the actual size is. cur_numrows and cur_numcols store the
   current number of rows and columns, respectively. */

cur_numcols = CPXgetnumcols (env, lp);
cur_numrows = CPXgetnumrows (env, lp);

/* Allocate space for basis and solution */

cstat = (int *) malloc (cur_numcols*sizeof(int));
rstat = (int *) malloc (cur_numrows*sizeof(int));
x = (double *) malloc (cur_numcols*sizeof(double));

if ( cstat == NULL || rstat == NULL || x == NULL ) {
    fprintf (stderr, "No memory for basis statuses.\n");
    goto TERMINATE;
}

/* If CPXgetbase causes an error, we don't want to see that error
   message on the screen. So turn off the screen indicator for
   this call, and turn it back on afterwards. */

CPXsetintparam (env, CPX_PARAM_SCRIND, CPX_OFF);
status = CPXgetbase (env, lp, cstat, rstat);
CPXsetintparam (env, CPX_PARAM_SCRIND, CPX_ON);

if ( status == CPXERR_NO_BASIS ) {
    printf ("No basis exists.\n");
    free_and_null ((char **) &cstat);
    free_and_null ((char **) &rstat);
}

```

```

    }
    else if ( status ) {
        fprintf (stderr, "Failed to get basis. error %d.\n", status);
        goto TERMINATE;
    }

    status = CPXgetx (env, lp, x, 0, cur_numcols-1);
    if ( status ) {
        fprintf (stderr, "Failed to obtain primal solution.\n");
        goto TERMINATE;
    }

    /* Write out the solution to a text or binary file given by argument 3*/

    status = CPXwritesol(env, lp, argv[3], NULL);

    /* for (j = 0; j < cur_numcols; j++) {
        printf ( "Column %d: Value = %17.10g", j, x[j]);
        if ( cstat != NULL ) {
            switch (cstat[j]) {
                case CPX_AT_LOWER:
                    basismsg = "Nonbasic at lower bound";
                    break;
                case CPX_BASIC:
                    basismsg = "Basic";
                    break;
                case CPX_AT_UPPER:
                    basismsg = "Nonbasic at upper bound";
                    break;
                case CPX_FREE_SUPER:
                    basismsg = "Superbasic, or free variable at zero";
                    break;
                default:
                    basismsg = "Bad basis status";
                    break;
            }
            printf (" %s", basismsg);
        }
        printf ("\n");
    }
    */

TERMINATE:

    /* Free up the basis and solution */

    free_and_null ((char **) &cstat);
    free_and_null ((char **) &rstat);
    free_and_null ((char **) &x);

    /* Free up the problem, if necessary */

    if ( lp != NULL ) {
        status = CPXfreeprob (env, &lp);
    }

```

```

    if ( status ) {
        fprintf (stderr, "CPXfreeprob failed, error code %d.\n", status);
    }
}

/* Free up the CPLEX environment, if necessary */

if ( env != NULL ) {
    status = CPXcloseCPLEX (&env);

    /* Note that CPXcloseCPLEX produces no output,
       so the only way to see the cause of the error is to use
       CPXgeterrorstring. For other CPLEX routines, the errors will
       be seen if the CPX_PARAM_SCRIND indicator is set to CPX_ON. */

    if ( status ) {
        char errmsg[1024];
        fprintf (stderr, "Could not close CPLEX environment.\n");
        CPXgeterrorstring (env, status, errmsg);
        fprintf (stderr, "%s", errmsg);
    }
}

return (status);
} /* END main */

```

```

/* This simple routine frees up the pointer *ptr, and sets *ptr to NULL */

```

```

#ifdef CPX_PROTOTYPE_MIN
static void
free_and_null (char **ptr)
#else
static void
free_and_null (ptr)
char **ptr;
#endif
{
    if ( *ptr != NULL ) {
        free (*ptr);
        *ptr = NULL;
    }
} /* END free_and_null */

```

```

#ifdef CPX_PROTOTYPE_MIN
static void
usage (char *progname)
#else
static void
usage (progname)

```

```

char *programe;
#endif
{
    fprintf(stderr,"Usage: %s filename algorithm\n", programe);
    fprintf(stderr,"  where filename is a file with extension \n");
    fprintf(stderr,"    MPS, SAV, or LP (lower case is allowed)\n");
    fprintf(stderr,"  and algorithm is one of the letters\n");
    fprintf(stderr,"    p or o  primal simplex (CPXprimopt)\n");
    fprintf(stderr,"    d or t  dual simplex  (CPXdualopt)\n");
    fprintf(stderr,"    b      barrier      (CPXbaropt)\n");
    fprintf(stderr,"    h      barrier with crossover (CPXhybbaropt)\n");
    fprintf(stderr,"    n      network simplex (CPXhybnetopt)\n");
    fprintf(stderr," Exiting...\n");
} /* END usage */

```

tt2leon for Metamodel Interface

```
#!/usr/local/bin/perl

srand; #initializes the random number generator

@days_to_degrade = (2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8);

open(IN, "SIMU90");

$degrade = "no"; #this tell us if the LP was degraded

$maxst{2009} = "2";
$maxst{2010} = "2";
$maxst{2011} = "12";
$maxst{2012} = "12";
$maxst{2013} = "14";
$maxst{2014} = "2";
$maxst{2015} = "1";
$maxst{2016} = "5";
$maxst{2017} = "1";

$numsectors = 31; #number of sectors in aij matrix -1 goes here

while (<IN>) {
    /\("[^"]+"\)/ && ($x = $1) =~ s/ //g;
    s(/\("[^"]+"\)/$x/;

    ($f0,$f1,$f2,$f3,$f4,$f5,$f6,$f7) = split;

    if ($f0 =~ /^TIME/) {
        chop($f1); chop($f1); chop($f1);
        $time = $f1;
    }

    elsif ($f0 =~ /^UNIT/) {
        $strengths{$f3} += $f5;
        $cmdid = $f3;
        $unit{$f1} = $f5;
        $unitid = $f1;
    }

    elsif ($f0 =~ /^UNAMMO/) {
        $ammo{$cmdid} += $f3;
    }
}
```



```

    }
    elsif ($f0 =~ /^UNBULK/) {
        $bulk{$cmdid} += $f3;
    }
    elsif ($f0 =~ /^UNPOL/) {
        $pol{$cmdid} += $f3;
    }

    elsif ($f0 =~ /^UNWATER/) {
        $water{$cmdid} += $f3;
    }

    elsif ($f0 =~ /^UNEQ/) {
        if ($f2 == 2100) {
            $tank{$cmdid} += $f6;
        }
        elsif ($f2 == 2200) {
            $apc{$cmdid} += $f6;
        }
        elsif ($f2 == 2300) {
            $helo{$cmdid} += $f6;
        }
        elsif ($f2 == 2400) {
            $harty{$cmdid} += $f6;
        }
        elsif ($f2 == 2500) {
            $larty{$cmdid} += $f6;
        }
        elsif ($f2 == 2600) {
            $inf{$cmdid} += $f6;
        }
        elsif ($f2 == 2700) {
            $adgun{$cmdid} += $f6;
        }
        elsif ($f2 == 2710) {
            $sfsam{$cmdid} += $f6;
        }
        elsif ($f2 == 2720) {
            $mobsam{$cmdid} += $f6;
        }
    }

    elsif ($f0 =~ /^SQAMMO/) {
        $sqammo{$f1} += $f3;
    }

    elsif ($f0 =~ /^SQPOL/) {
        $sqpol{$f1} += $f3;
    }

```

```

    }

    # this section grabs the Squadron IDs and total sorties flown in the last period
    elsif ($f0 =~ /^SQPLAN/) {
        $strengths{$f1} += $f3;
    }

    if ($f6 =~ /^"ON/) {
        $cons{$f3} += $f5;
    }

    if ($f0 =~ /^SQPLAN/) {
        if ($f2 =~ /^STI/) {
            $cons{$f1} += $f4;
        }

        elsif ($f2 =~ /^OCA/) {
            $cons{$f1} += $f4;
        }

        elsif ($f2 =~ /^INT/) {
            $cons{$f1} += $f4;
        }

        $f10 = "$f1 + $f2";
        $plansort{$f10} = $f3;
        $planstrg{$f1} += $f4;
        $mission{$f1} += $f2;
        $sqnid = $f1;
        $sorties{$f2} = $f3;
    } # end if for the squadron plan portion

} # end of while loop


if ($time == 1){
    open (I, ">i.dat");
    $degcomm[0] = $days_to_degrade[rand(@days_to_degrade)];
    print I "$degcomm[0]";
}

```

```

else{

    open( I, "i.dat");

while (<I>) {
    @degcomm = split;
    }

}

close(I);

$degcomm[0] = 2.5; # remove this line for stochastic draws

# this creates the targets for the strat effects
#if($time <=2){
    $Cmnd = 170;
#
#
#else {
    $Cmnd = 70;
#
#
if($time >= $degcomm[0]){
    $Comm = 50;
}
else
{
    $Comm = 150;
}

print " Comm value is $Comm";

rename("SIMU90", "SIMU90.$time"); #this renames and closes the input file to save the data
close (IN);
$time = $time - .5;
if ($time > 1){
    rename("SIMU91","SIMU91.$ptime"); #this renames the last SIMU91 file with last periods ext
}

# this section makes the command strength and consumption into % of total possible strength

foreach $key (keys %maxst) {
    $strengths{$key} = ($strengths{$key} / $maxst{$key}) *100;
    $strengths{$key} = substr($strengths{$key}, 0, 6);
}

foreach $key (keys %maxst) {
    $cons{$key} = ($cons{$key} / $maxst{$key}) *100;
    $cons{$key} = substr($cons{$key}, 0, 6);
}

# this is the beginning of building an lp using the data retrieved above

```

```

# the following lines puts the keys into the proper order expected by S3I's program

@commheads = (2009..2017);
@unitheads = (2101..2112, 2201..2212, 2301..2309, 2313..2317, 2401..2403, 2034..2037, 2039, 2040, 2044,
2045, 2046, 2047);
@sqnheads = (22901, 22101..22106, 20101..20105, 22501, 22502, 22301..22304, 20000, 29902..29904, 29901);
@missheads = ("DCA", "ODCA", "HVAA", "BARCAP", "FSWP", "AIRES", "STI", "CAS", "BAI",
"INT", "OCA", "OTBM", "DTBM", "DSEAD", "SSUP", "CSUP", "ESCSUP", "SJAM", "CJAM",
"ESCJAM", "RECCE", "SREC", "AEW", "AAR", "LIFT", "XXXX", "RESERVE");

if ($time == 1.000) { # for time 1.000 we want to just return a default file to begin the SIM
    $write = &make_default_SIMU91;
}
else {

open(LP, ">equation.lp"); # this is the file where the lp is written

# This array will act as the tool to build the equations of the lp

@heads = sort by_mostly_numeric keys %strengths; #when building the aij matrix everything is sorted

print LP "maximize\n\n"; # this sets the objective function of minimizing the difference between
consumption

    for ($i = 1; $i <= $numsectors + 1; $i++) {

        if ($i == 1){
            print LP "Y$i";
        }
        else {
            print LP " + Y$i";
        }
    }

    print LP " + X33 + X34"; # these are added for the strattgt

print LP "\n \nst \n\n";

open(AIJS, "/home/nisa2/students/asnodgra/Baseline/aij.txt"); #opens a file of aij's

$i = 0; # serves as a counter

while (<AIJS) {

    @aij = split;
    $boolean = "0"; # this will be tested to see if it is the start of the line
    $l = $i+1; # this transforms the array count to the variable count

    $net = (1- $aij[$i]); #provides the aii multiplier

```

```

for ($j = 0; $j <= $numsectors; $j++) {
    $m = $j+1;

    if ($i == $j && $boolean == 0) {
        print LP "$net X$m";
        $boolean++;
    }
    elseif ($i == $j && $boolean != 0) {
        print LP " + $net X$m";
        $boolean++;
    }
    elseif ($aij[$j] > 0) {
        print LP " - $aij[$j] X$m";
        $boolean++;
    }
} # closes the for j loop

print LP " - Y$1 >= 0\n";

$i++;

} # this closes the while statement

#added for Strattgt
print LP "X34 - .27 X3 - .27 X4 - .27 X5 - .10 X8 >= 0\n";
print LP "X33 - .65 X4 >= 0\n";

close (AIJS);

#open(PIJS, "/home/nisa2/students/asnodgra/Baseline/pij.txt");

#$counter = 1;

#while (<PIJS>){ # this will move us from resource type to resource type

#    @pij = split;

#    for($i = 0; $i <= $numsectors; $i++) { # this takes us across the sectors

#        if ($pij[$i] >= .01){
#            $l = $i + 1;
#            $key = $heads[$i]; #this gets the command/sq key for the type of resource
#
#            if ($counter == 1) {
#                print LP "$pij[$i] X$l <= $sqammo{$key}\n";
#            }
#            elseif ($counter == 2) {
#                print LP "$pij[$i] X$l <= $sqpol{$key}\n";
#            }
#            elseif ($counter == 3) {
#                print LP "$pij[$i] X$l <= $ammo{$key}\n";
#            }
#        }
#    }
#}

```

```

    }
    print LP "0 <= X33 <= $Comm\n";
    print LP "0 <= X34 <= $Cmnd\n";

close (PIJS);
close (LP);

$totalcons = $totalcons + $Comm + $Cmnd; # this was added to account for the strat targets

print "Total Consumption is $totalcons";

system("/home/nisa2/students/asnodgra/CplexLeon2/solver equation.lp p itworks.txt");

open (SOL, "itworks.txt");

@newout = (0..32); #this gives us default values for the datafile
$j = "0"; #this is used to count the number of variables collected
while (<SOL>) {

    @vals = split;

    if ($vals[0] =~ /^A/) { # Cplex put an A at the beginning of some lines
        $vals[1] = $vals[2];
        $vals[3] = $vals[7];
    }
    $what = substr($vals[1],0,1);

    if ($vals[1] =~ /^VALUE/ && $vals[2] >= int($totalcons)) {

        $writefile = &make_default_SIMU91;
        last;
    }
    elsif ($what =~ /^X/) {

        $i = 1;
        until ($vals[1] =~ /^bX$i\b/){
            $i++;
        }
        $newout[$i] = $vals[3];
        $j++;
    }
    $count = $numsectors + 1;
    if ($j == $count) {

        $makeest = &make_strengths_nonzero;
        $writefile = &make_degrade_SIMU91;
        $degrade = "yes";
    }
}

```

```

} # end reading the equation.lp

close(SOL);
rename("itworks.txt", "itworks.$time");
rename("equation.lp", "equation.$time");

open (DAT, ">>datafile.txt"); #this records when a degrade has been passed
$i = 1;
for $key(@heads) {
    print DAT "$strengths{$key}    $newout[$i]    ";
    $i++;
}

    print DAT "$degcomm[0] nocmnddeg $time $degrade\n";
} # end the else statement for things to do after time 1

# this is the beginning of the SIMU91 file

sub make_degrade_SIMU91 {

$outfile = "SIMU91";

open(OUT, ">$outfile");

    for $key (@unitheads) {

        if ($key == 2046 || $key == 2047){
            $out = $newout[1] / $strengths{2009};
            $out = substr($out,0,4);
            print OUT "UNIT $key $out\n";
        }
        elseif ($key == 2044 || $key == 2045){
            $out = $newout[2]/$strengths{2010};
            $out = substr($out,0,4);
            print OUT "UNIT $key $out\n";
        }
        elseif ($key >= 2101 && $key <= 2112){
            $out = $newout[3]/$strengths{2011};
            $out = substr($out,0,4);
            print OUT "UNIT $key $out\n";
        }
        elseif ($key >= 2201 && $key <= 2212){
            $out = $newout[4]/$strengths{2012};
            $out = substr($out,0,4);
            print OUT "UNIT $key $out\n";
        }
        elseif ($key >= 2301 && $key <= 2309) {
            $out = $newout[5]/$strengths{2013};
            $out = substr($out,0,4);
        }
    }
}

```

```

print OUT "UNIT $key $out\n";
}
elseif ($key >= 2313 && $key <= 2317) {
$out = $newout[5]/$strengths{2013};
$out = substr($out,0,4);
print OUT "UNIT $key $out\n";
}
elseif ($key == 2034 || $key == 2035){
$out = $newout[6]/$strengths{2014};
$out = substr($out,0,4);
print OUT "UNIT $key $out\n";
}
elseif ($key == 2036){
$out = $newout[7]/$strengths{2015};
$out = substr($out,0,4);
print OUT "UNIT $key $out\n";
}
elseif ($key == 2039 || $key == 2040 || ($key >= 2401 && $key <= 2403)){
$out = $newout[8]/$strengths{2016};
$out = substr($out,0,4);
print OUT "UNIT $key $out\n";
}
elseif ($key == 2037){
$out = $newout[9]/$strengths{2017};
$out = substr($out,0,4);
print OUT "UNIT $key $out\n";
}
} # ends the unit for statement

for $key (@sqnheads) {

    for $key2 (@missheads) {
        $f11 = "$key + $key2";

        if ($key == 20000){
            $out = $newout[10]/$strengths{$key};
            $sorties = $out * $plansort{$f11};
            $sorties = int($sorties + .5);
            print OUT "SQN $key $key2 $sorties\n";
        }
        elseif ($key == 20101){
            $out = $newout[11]/$strengths{$key};
            $sorties = $out * $plansort{$f11};
            $sorties = int($sorties + .5);
            print OUT "SQN $key $key2 $sorties\n";
        }
        elseif ($key == 20102){
            $out = $newout[12]/$strengths{$key};
            $sorties = $out * $plansort{$f11};
            $sorties = int($sorties + .5);
            print OUT "SQN $key $key2 $sorties\n";
        }
        elseif ($key == 20103){

```



```

Sout = $newout[13]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 20104){
Sout = $newout[14]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 20105){
Sout = $newout[15]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22101){
Sout = $newout[16]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22102){
Sout = $newout[17]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22103){
Sout = $newout[18]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22104){
Sout = $newout[19]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22105){
Sout = $newout[20]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22106){
Sout = $newout[21]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elsif ($key == 22301){

```

```
$out = $newout[22]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22302){
$out = $newout[23]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22303){
$out = $newout[24]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22304){
$out = $newout[25]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22501){
$out = $newout[26]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22502){
$out = $newout[27]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 22901){
$out = $newout[28]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 29901){
$out = $newout[29]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 29902){
$out = $newout[30]/$strengths{$key};
$sorties = $out * $plansort{$f11};
$sorties = int($sorties + .5);
print OUT "SQN $key $key2 $sorties\n";
}
elseif ($key == 29903){
```

```

        $out = $newout[31]/$strengths{$key};
        $sorties = $out * $plansort{$f11};
        $sorties = int($sorties + .5);
        print OUT "SQN $key $key2 $sorties\n";
    }
    elsif ($key == 29904){
        $out = $newout[32]/$strengths{$key};
        $sorties = $out * $plansort{$f11};
        $sorties = int($sorties + .5);
        print OUT "SQN $key $key2 $sorties\n";
    }

    } # ends the for routine for missions
    } # ends the for routine for squadrons

    close(OUT);
} # ends the subroutine

sub make_default_SIMU91 {

    $outfile = "SIMU91";

    open(OUT, ">$outfile");

    for $key (@unitheads) {
        print OUT "UNIT $key 1.0\n";
    }

    for $key (@sqnheads) {
        for $key2 (@missheads) {
            $f11 = "$key + $key2";
            print OUT "SQN $key $key2 $plansort{$f11}\n";
        }
    }
    close (OUT);

} # this is the end of the section that prints a default SIMU91 file with no degrades

sub make_strengths_nonzero {

    for $key (@commheads) {

        if ($strengths{$key} == 0) {
            $strengths{$key} = 1; #this keeps me from dividing by zero
        }

    }

}

```

```

    }
for $key (@sqnheads) {
    if ($strengths{$key} == 0) {
        $strengths{$key} = 1; #this keeps me from dividing by zero
    }
} # end of subroutine

```

```

sub by_mostly_numeric {
    ($a <=> $b) || ($a cmp $b);
}

```